THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité : **Informatique (EDITE)**

Présentée par : **M. Jean-Baptiste MOURET**

Pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

# Pressions sélectives multiples pour l'évolution de réseaux de neurones destinés à la robotique

Thèse dirigée par **Jean-Arcady MEYER et Stéphane DONCIEUX**
soutenue le 5 décembre 2008
devant le jury composé de :

| | | |
|---|---|---|
| Pr. Philippe BIDAUD | (ISIR, Univ. Pierre et Marie Curie) | Examinateur |
| Dr. Stéphane DONCIEUX | (ISIR, Univ. Pierre et Marie Curie) | Invité |
| Pr. Yves DUTHEN | (IRIT, Univ. Toulouse 1) | Rapporteur |
| Dr. Jean-Arcady MEYER | (ISIR, Univ. Pierre et Marie Curie) | Directeur de thèse |
| Pr. Hélène PAUGAM-MOISY | (LIRIS, Univ. Lumière Lyon 2) | Examinatrice |
| Pr. Patrice PERNY | (LIP6, Univ. Pierre et Marie Curie) | Examinateur |
| Dr. Marc SCHOENAUER | (TAO, INRIA-Futurs) | Rapporteur |

## RÉSUMÉ

De nombreux travaux montrent que les algorithmes évolutionnistes peuvent fournir des solutions efficaces pour beaucoup de problèmes en robotique. Cependant, ils peinent à mettre au point des artefacts complexes lorsque la fitness est insuffisante pour guider explicitement le processus. Supposant que la complexité des êtres vivants provient en partie de la multiplicité des pressions sélectives, nous proposons la création de telles pressions pour l'évolution de neuro-contrôleurs à l'aide d'algorithmes évolutionnistes multiobjectifs.

Nous commençons par décrire comment des hypothèses sur des étapes intermédiaires peuvent être exploitées en définissant un problème d'optimisation multiobjectif où chaque objectif correspond à une étape.

L'ajout d'un objectif poussant à l'exploration du voisinage des solutions déjà obtenues constitue une autre possibilité pour palier au manque de gradient de sélection. La comparaison de différentes méthodes pour appliquer ce concept à l'évolution de réseaux de neurones montre l'intérêt de maintenir la diversité parmi les comportements des solutions, et non parmi les génotypes ou les phénotypes.

Enfin, nous montrons que les exaptations peuvent être favorisées via des pressions sélectives sur des modules phénotypiques reliés à des modules génotypiques.

Ces méthodes ont été testées sur l'évolution de réseaux de neurones calculant des fonctions logiques et sur la génération de neuro-contrôleurs pour un robot phototrope. Elles peuvent être appliquées à un large spectre de problèmes de robotique évolutionniste, du pilotage de robots à l'obtention de contrôleurs rythmiques.



### MOTS CLEFS

algorithmes évolutionnistes multiobjectifs ; exaptation ; gradients de sélection ; robotique évolutionniste ; réseaux de neurones ; modularité.

## ABSTRACT

### MULTIPLE SELECTION PRESSURES FOR THE EVOLUTION OF NEURAL NETWORKS IN ROBOTICS

Evolutionary algorithms have been successfully used to generate controllers for many robots. However, they struggle to design complex artifacts when the fitness is unable to explicitly guide the process. In this thesis, we draw the hypothesis that these problems originate from the use of a single selection pressure, whereas living organisms are subject to many ones. We investigate here the use of multiobjective evolutionary algorithms to create such multiple gradients in order to evolve neuro-controllers.

We first describe how hypotheses about potential intermediate steps can be used by defining a multiobjective optimization problem in which each objective corresponds to a sub-task.

In the lack of any selection pressure, it is also possible to add an objective which encourages an efficient exploration of the neighborhood of current candidate solutions. We consider several possibilities to instantiate this concept for the evolution of neural networks and we conclude that maintaining the diversity of the behaviors, instead of the diversity of the genotype or the phenotype, is an efficient way to override the deceptiveness of a fitness function.

Last, we show that exaptations can be favored by applying a selection pressure on some modules of the generated neural-networks, possibly linked to genotypic modules.

We tested these methods on the evolution of neural networks to compute a Boolean function and to control a light-seeking robot. They seem to be applicable to a wide range of evolutionary robotics problems, from complex locomotion to behavior control.



### KEYWORDS

multiobjective evolutionary algorithms; selection gradients; exaptation; evolutionary robotics; neural networks; modularity.

## LABORATOIRE D'ACCUEIL


ISIR - Institut des Systèmes Intelligents et de Robotique
4 Place Jussieu
75252 Paris Cedex
Encadrants : Jean-Arcady Meyer et Stéphane Doncieux

# REMERCIEMENTS

LA PAGE de remerciements peut paraître bien futile dans un document à vocation purement scientifique. Après tout, il s'agit simplement d'une liste de noms saupoudrée de quelques mots de contexte ; rien d'intéressant ! Pourtant, je ne pourrais énumérer les textes de remerciement que j'ai lu avec intérêt dans les thèses de mes congénères car, à la lecture de ces quelques paragraphes, j'ai pu jeter un regard la personnalité du rédacteur. Qui est cette personne dont je m'apprête à lire la prose ? Quel est son univers personnel ? De cette partie du manuscrit, la seule à vocation personnelle, on ne peut extraire beaucoup d'information. Mais cette petite fenêtre ouverte sur une vie se déguste parfois comme un petit bonbon acidulé : sans intérêt gastronomique mais avec plaisir. Je me prête donc à l'exercice, prêt à satisfaire votre curiosité mais, surtout, pour arracher un sourire à ceux que je n'oublierais pas.

Commençons par le plus simple : je tiens à remercier mes parents qui n'ont jamais cessé de stimuler ma soif de connaissance. Ils m'ont offert les moyens d'être curieux en me traînant avec mes petits frères au fond des musées et en m'incitant à bricoler ordinateurs et robots. Merci.

Je glisse ce même mot dans l'enveloppe destinée à Céline, qui ne cesse de m'encourager et à même accepté de relire une partie de ce manuscrit.

J'ai toujours été impressionné par la confiance qu'ont placée en moi les membres de l'AnimatLab et notamment, bien sûr, Stéphane Doncieux et Jean-Arcady Meyer. Ils ont su se démener pour me permettre d'obtenir financements et matériel tout en étant toujours prompts à répondre à mes interrogations scientifiques. Stéphane, Jean-Arcady, Agnès et Olivier : merci.

Dans tous les labos et dans toutes les écoles, on croise quantité de gens intéressants, on déguste des bières en déconstruisant le monde et, parfois, on discute même de science (mais tard, après beaucoup d'alcool). Je ne saurais être exhaustif, mais je souhaite au moins déposer un merci sous la porte (virtuelle) des Happycoders, du SDEN et du LRDE et, bien sûr, de l'ensemble des doctorants et post-doctorants de l'AnimatLab.

Enfin, je remercie mon jury qui a accepté de prendre le temps de se pencher sur mes élucubrations scientifiques malgré des emplois du temps que je sais chargés.

TABLE DES MATIÈRES

# 1

# INTRODUCTION

> « In considering transitions of organs, it is so important to bear in mind the probability of conversion from one function to another [...]. »
>
> Charles Darwin.
> *The Origin of Species*, page 191.
> 1859.

LES ROBOTS MOBILES et les drones sont chaque jour plus utilisés, du nettoyage des appartements à la surveillance des zones de conflit armé. Quelle que soit leur utilisation, ils sont immergés dans un environnement complexe et dynamique. À l'échelle d'un robot, le sol d'un appartement, par exemple, est souvent jonché d'obstacles à la position impossible à prévoir. De manière semblable, les drones doivent voler dans une masse d'air sans-cesse agitée par d'invisibles mouvements de convection et de nombreuses turbulences tout autant difficile à prévoir. De telles situations rendent souvent inefficaces les notions classiques en robotique d'analyse, de cartographie et de planification car elles requièrent une connaissance détaillée de l'environnement du robot.

L'observation de la nature souffle une autre voie de recherche : les oiseaux jouent avec l'atmosphère sans en comprendre le fonctionnement, les rats trouvent leur repas dans des réseaux de galeries sans tracer de plans et les insectes repèrent leur nourriture dans nos placards sans avoir besoin du concept de garde-manger. Comment font-ils ? Chaque espèce s'appuie sur des solutions uniques mais toutes ont été façonnées par l'évolution. Aveugle face aux mécanismes qui régissent notre monde, la sélection naturelle permet d'exploiter la variabilité entre les individus pour les adapter à leur environnement. En fermant ainsi les yeux, en n'exploitant ni analyse, ni calcul, ni plan, l'évolution a ainsi pu emprunter des raccourcis hors de portée de la vision analytique sous-jacente à l'ingénierie et à la science. L'imitation du processus évolutionniste, dont beaucoup de détails restent mystérieux, n'est certainement pas aisée. Elle n'est probablement pas non plus la voie la plus courte pour créer des robots plus adaptés à leur environnement. Néanmoins, l'observation de la nature nous démontre quotidiennement la capacité de l'évolution à mener à des organismes performants ; s'en inspirer revient donc à choisir un chemin en étant assuré de son issue mais sans connaître les obstacles qu'il traverse.

La généralité et les possibilités du processus évolutionniste ont inspirés la recherche en informatique dès ses débuts. Des développements parallèles ont mené à différentes méthodes d'optimisation regroupées sous le terme d'*algo-*

*rithme évolutionnistes*[1] (*evolutionary algorithms*, voir par exemple de Jong (2002)). Dans ces abstractions du processus évolutionniste, des opérateurs de mutation et de croisement modifient les individus d'une population de solutions candidates. Ces dernières sont mises en concurrence par l'entremise d'une fonction évaluant leurs performances, baptisée *fonction de fitness*[2], qui influe sur le taux de reproduction des individus. L'utilisation de ces algorithmes pour la mise au point de la morphologie et de l'architecture de contrôle de robots a mené au domaine de la *robotique évolutionniste* (Cliff et al., 1993b; Meyer et al., 1998; Meyer, 1998; Nolfi et Floreano, 2001). Pour créer des solutions originales et comprendre les conditions de leur émergence, le *credo* de cette approche est la minimisation de la connaissance apportée au processus afin de lui laisser le maximum de liberté. Dans cette optique, beaucoup de travaux se sont appuyés sur la généralité des réseaux de neurones pour faire évoluer des neuro-contrôleurs capables, par exemple, de permettre à un robot hexapode de marcher, éviter les obstacles et suivre un gradient (Kodjabachian et Meyer, 1997), de piloter un dirigeable lenticulaire (Doncieux et Meyer, 2003a), de guider une fusée simulée (Gomez et Miikkulainen, 2003) ou encore d'adapter le battement d'ailes d'un robot-oiseau simulé (Mouret et al., 2006). Ces exemples cachent cependant de nombreuses applications infructueuses en robotique évolutionniste. Obtenir un neuro-contrôleur pour un robot en utilisant un processus évolutionniste peu guidé est toujours un défi. En créer un pour des comportements arbitrairement complexes, comme ceux nécessitant une mémoire de plus de quelques bits, semble inaccessible aux méthodes actuelles.

Pourtant, bien que sculptés par l'évolution, les organismes vivants sont d'une indéniable complexité. Dans chacun d'eux, de nombreux organes eux-mêmes composés de nombreuses cellules interagissent selon un réseau particulièrement complexe. On ne peut aussi que s'émerveiller des capacités du système nerveux des animaux, même des plus simples, alors que la science ne fait qu'effleurer les mystères de la cognition. Comment des systèmes si complexes ont-ils pu émerger uniquement par sélection naturelle ? Cette interrogation reflète la fameuse critique de Mivart à Darwin (voir Gould (1985)). Selon Mivart, on peut comprendre comment des structures complexes sont préservées, voire améliorées, en utilisant uniquement la sélection naturelle. Mais comment expliquer la création à partir de rien d'une structure aussi élaborée que l'aile d'un oiseau si l'évolution doit s'appuyer sur une longue séquence d'étapes, chacune favorisées par la sélection naturelle ? En d'autres termes, à quoi sert la moitié d'une aile puisqu'elle ne permet pas de voler ? En un sens, les difficultés en robotique évolutionnistes peuvent illustrer l'argument de Mivart : dans de nombreux exemples (Harvey et al., 1994; Kodjabachian et Meyer, 1997; Urzelai et Floreano, 1999), tous les individus des premières générations obtiennent la fitness minimale, aucune pression sélective n'est donc appliquée ;

1 ou « algorithmes évolutionnaires ».

2 On emploiera l'anglicisme « fitness » dans ce mémoire faute de terme équivalent couramment admis dans le domaine en français. On parle parfois de fonction d'adaptation.

par conséquent, aucun résultat significatif ne peut être trouvé. Le processus de sélection tel qu'il est envisagé semble alors insuffisant pour réussir la tâche abordée. Plus généralement, les méthodes évolutionnistes se révèlent très efficaces pour le perfectionnement itératif de solutions mais elles semblent se heurter à un mur dès que des étapes intermédiaires doivent être maîtrisées avant de pouvoir s'intéresser aux capacités récompensées par la fitness.

Darwin consacre un chapitre de la dernière édition de l'*Origine des espèces* (1872) à répondre aux critiques émises par Mivart. Il y explique qu'une supposition cachée dans le raisonnement de son opposant doit être écartée : la continuité fonctionnelle. Il est certes difficile de voler avec la moitié d'une aile, mais pourquoi cette demi-aile devrait elle nécessairement servir pour le vol ? Les ailes pourraient, par exemple, avoir été originellement adaptées pour courir plus vite et attraper mieux les insectes (voir par exemple Shipman (1998)) ou gravir des plans inclinés (Dial, 2003). Darwin nomme ce décalage de fonction la *préadaptation*, un terme maintenant remplacé par celui d'*exaptation* (Gould et Vrba, 1982), moins connoté. Au delà de l'apparition des ailes, les biologistes ont ainsi documenté de nombreux cas d'exaptation dans l'histoire du vivant. Ce concept apparaît fondamental pour la compréhension de développements aussi déterminants que l'apparition des os, initialement adaptés au stockage de certains minéraux, ou celle des pattes chez les tétrapodes, originellement utiles dans l'eau mais ayant permis la conquête de la terre.

Si de tels changements dans les fonctions des organes sont indispensables pour expliquer l'évolution des êtres vivants, on ne peut que s'étonner de leur apparente absence des expériences robotique évolutionniste. Dans la nature comme dans les algorithmes évolutionnistes, la fitness modifie les taux de sélection, favorisant les individus les plus adaptés. Où se situe la différence ? Dans cette thèse, nous formulons l'hypothèse que l'*exaptation n'apparaît pas dans les algorithmes évolutionnistes car la fonction de fitness ne fournit qu'un seul gradient de sélection*. En d'autre termes, la simplicité des fonctions de fitness utilisées ne permet pas aux solutions obtenues de s'appuyer sur des solutions intermédiaires adaptées à d'autres fonctions, simplement car celles-ci n'existent pas dans le processus. À l'inverse, les êtres vivants sont soumis à de nombreuses pressions de sélection du fait de la richesse de leurs interactions avec l'environnement et de la complexité des organismes. Par exemple, un rat peut augmenter sa fitness en se déplaçant plus vite, et ainsi fuir le danger plus efficacement, mais aussi en perfectionnant son acuité auditive, en résistant mieux aux maladies ou encore en améliorant ses capacités de mémorisation. Un rat plus agile que ses congénères accroîtra ses chances de survie, et donc de reproduction ; mais un de ses frères doté de très bon yeux aussi. On ne peut savoir quelles fonctions serons exaptées dans le futur de l'espèce, mais leur multiplicité crée autant de gradients de sélection sur lesquels l'évolution pourra s'appuyer.

Conscients des difficultés pour la complexification des organismes induites par l'utilisation d'une fonction de fitness explicite, de nombreux auteurs (par

exemple, Ray (1991); Ackley et Littman (1992); Elfwing et al. (2005)) ont proposé de modéliser plus finement le processus de sélection naturelle en le situant dans un environnement. Les robots, simulés ou non, habitant un tel univers artificiel interagissent et, à cette occasion, peuvent se reproduire. Les ressources nécessaires à la survie étant limitées, la propagation d'un gène dans la population dépend de nombreux facteurs, du taux de reproduction à la capacité à trouver la nourriture. Surtout, le compromis entre ces facteurs n'est pas fixé par l'expérimentateur mais émerge de l'interaction entre les organismes artificiels. La fitness est alors *implicite* et *intrinsèque* aux individus. La pression sélective pour chaque caractéristique dépend de l'ensemble des individus, elle change donc dans le temps en fonction des capacités développées par les individus. L'issue de l'évolution est alors plus ouverte que dans les méthodes évolutionnistes classiques, le système évoluant sans but défini. En abandonnant ainsi la notion d'objectif, de tels écosystèmes artificiels s'éloignent des méthodes d'optimisation pour s'approcher de la « vie artificielle ». Si leur intérêt est indéniable pour la compréhension de dynamiques de l'évolution, leur nature implicite les rend difficile à utiliser pour résoudre des problèmes pratiques de robotique. Mettre au point une fonction de fitness correspondant à la réussite d'une tâche peut parfois être difficile ; mais concevoir un écosystème dans lequel survivre implique implicitement la réalisation d'une fonction donnée semble être une tâche réservée aux démiurges les plus doués ! Pour obtenir par évolution des robots complexes mais aussi utiles, des méthodes plus explicites semblent donc plus adaptées.

La présence d'un gradient de sélection n'est pas suffisante pour que les rats puissent améliorer leur acuité auditive ; il est aussi nécessaire que les modifications génotypiques affectant leur audition aient peu d'influence néfaste sur leurs autres capacités, et notamment celles nécessaires à la survie. Cette indépendance ou quasi-indépendance entre les gènes codant pour des caractères différents traduit la *modularité* de la correspondance entre le génotype et le phénotype (voir par exemple (Bonner (1988), p. 175)). Si les conditions d'apparition d'une telle organisation modulaire des êtres vivants sont loin d'être élucidées (voir Callebaut et Rasskin-Gutman (2005)), son existence apparaît indissociable du concept d'exaptation car elle relie le génotype aux gradients de sélection. De plus, en isolant ainsi les fonctions de chaque trait phénotypique, la modularité facilite la répétition d'une sous-partie au sein d'un même organisme. Un module dupliqué est alors susceptible d'être utilisé pour une autre fonction sans détériorer ses performances dans le contexte où il a été mis au point. L'importance de la modularité a été largement reconnue en robotique évolutionniste (Gruau, 1995; Calabretta et al., 1998; Doncieux et Meyer, 2004a; Reisinger et al., 2004) car elle pourrait augmenter la capacité des systèmes complexes à acquerir de nouvelles fonctions par des changements génétiques (Wagner, 2005). À notre connaissance, elle n'a cependant jamais été reliée explicitement à des gradients de sélection.

Si la modularité et la multiplicité des gradients de sélection semblent former des clefs importantes pour l'évolution de systèmes complexes, les outils méthodologiques et algorithmiques permettant de transcrire explicitement ces concepts à la robotique évolutionniste doivent encore être définis. Dans cette thèse, nous proposons d'utiliser les récents développements en optimisation qui ont mené aux *algorithmes évolutionnistes multi-objectifs* (voir Deb (2001)). Dans les algorithmes évolutionnistes classiques, les différents objectifs à optimiser sont agrégés en une unique fonction de fitness. Les gradients de sélection sont alors fondus ensemble lors de l'agrégation, qui définit l'importance relative de chacun d'eux. Selon cette approche, la pondération entre les objectifs peut, par exemple, amener le processus à considérer un rat particulièrement agile comme plus adapté qu'un autre au sens olfactif surdéveloppé. Les algorithmes multiobjectifs éliminent cette pondération en utilisant la notion d'optimum de Pareto : deux rats, chacun meilleur que l'autre pour un des objectifs, sont considérés comme étant de performance équivalente, alors qu'un de leur congénère moins bon sur les deux objectifs correspondra à une performance inférieure. Les algorithmes évolutionnistes multiobjectifs cherchent l'*ensemble des compromis* optimum au sens de Pareto et non plus une unique solution optimale. En refusant de regrouper tous les objectifs sous un seul et de viser un unique optimum, ces algorithmes introduisent donc des pressions de sélection multiples qui pourraient combler le manque de gradients de sélection observé en robotique évolutionniste.

## PROBLÈMATIQUE

En récapitulant les arguments que nous venons de développer, nous pouvons formuler les trois hypothèses suivantes :

1. l'un des principaux freins à l'émergence de solutions complexes en robotique évolutionniste est la présence d'un unique gradient de sélection ;
2. pour qu'un système complexe puisse suivre un gradient de sélection particulier, les traits phénotypiques à adapter doivent correspondre à un module du génotype ;
3. les algorithmes évolutionnistes multiobjectifs peuvent permettre de créer plusieurs gradients de sélection.

S'appuyant sur ces trois hypothèses, notre problématique prend la forme de la question : *comment exploiter l'évolution multiobjectif pour synthétiser des systèmes complexes ?*

Nous avons choisi d'appliquer nos propositions à l'évolution de réseaux de neurones pour le contrôle de robots mobiles, des systèmes complexes et beaucoup étudiés.

## PLAN DU MÉMOIRE

Le reste de ce mémoire est organisé en quatre chapitres.

Afin de cerner le contexte de ce travail, nous décrirons dans le premier chapitre les algorithmes évolutionnistes multiobjectifs, les liens entre évolution et modularité, l'évolution de réseaux de neurones et la robotique évolutionniste.

Le deuxième chapitre met en évidence les liens entre l'évolution par étapes, dite incrémentale, et l'optimisation multiobjectif. Ces considérations nous amèneront à présenter un premier exemple d'utilisation des algorithmes évolutionnistes multiobjectifs pour un problème de robotique mobile mettant en jeu de nombreuses étapes intermédiaires.

Nous consacrerons ensuite un chapitre à l'exploitation d'un gradient de sélection poussant à l'exploration. Les approches présentées seront testées sur un problème-jouet basé sur une fonction logique et sur un problème de robotique mobile.

Le quatrième chapitre décrira une méthode permettant d'associer les modules des solutions avec des gradients de sélection. Les résultats obtenus avec cette approche pour un problème simple seront analysés en détail.

Le dernier chapitre sera consacré à la mise en perspective du travail effectué. Dans un premier temps, nous décrirons comment appliquer les méthodes proposées dans cette thèse à différents problèmes de robotique évolutionniste. Nous analyserons dans un second temps les problèmes soulevés par les approches abordées dans ce mémoire et tenterons d'apporter quelques éléments de réponse.

# 2

# FONDATIONS

RÉSUMÉ. Ce chapitre présente un état de l'art succinct des différents domaines de recherche sur lesquels repose cette thèse. Nous commençons par présenter l'outil qui nous permettra de mettre en place plusieurs gradients de sélection, les algorithmes évolutionnistes multiobjectifs. Nous dressons ensuite un rapide panorama des relations entre l'évolution et la modularité dans le vivant afin de fournir une base de réflexion sur l'exploitation de modules dans l'évolution de réseaux de neurones. La section suivante est consacrée à la description des différentes méthodes existantes pour l'évolution de réseau de neurones, des codages directs aux approches modulaires. Nous présentons ensuite la robotique évolutionniste et l'utilisation de réseaux de neurones dans ce contexte. La dernière partie met en perspective cet état de l'art en dégageant les convergences entre les différents domaines abordés.

## 2.1 ALGORITHMES ÉVOLUTIONNISTES MULTIOBJECTIFS

### 2.1.1 *Algorithmes évolutionnistes*

EN PUBLIANT *L'Origine des espèces* (Darwin, 1859), Darwin expose pour la première fois sa théorie de l'évolution, désormais communément admise. Celle-ci repose sur les observations suivantes :

- Il existe au sein de chaque espèce de nombreuses variations, chaque individu étant différent.
- Les ressources naturelles étant finies, il naît rapidement plus d'être vivants que la nature ne peut en nourrir ; il en résulte une lutte pour l'existence (*struggle for life*) entre chaque organisme.
- Les individus survivants possèdent des caractéristiques qui les rendent plus aptes à survivre. Darwin baptise ce concept *sélection naturelle*.
- Les organismes survivants transmettent leurs avantages à leur descendance. l'accumulation au cours des générations des petites différences entre chaque branche généalogique crée de nouvelles espèces de plus en plus *aptes* à survivre.

Ces quatre principes de la théorie darwinienne ont été progressivement étendus pour incorporer les découvertes des lois de l'hérédité (voir Castle (1903)) puis de la structure de l'ADN (Crick et Watson, 1953) et, plus généralement, de

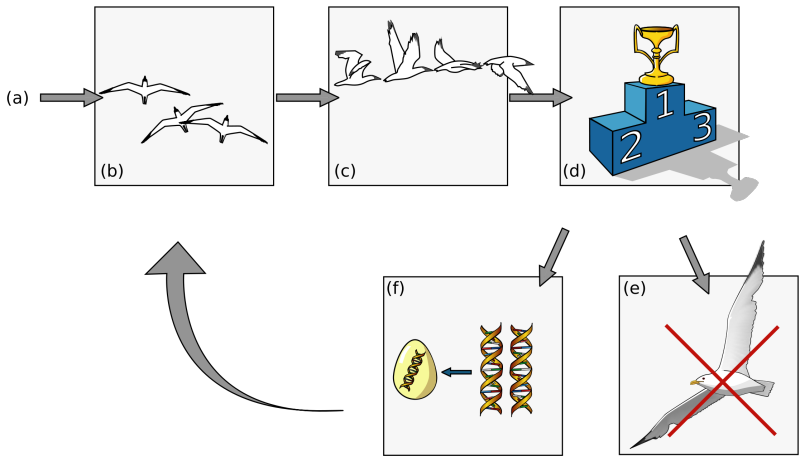



FIG. 2.1.: Principe général d'un algorithme évolutionniste. (a) Une population de solutions candidates est générée aléatoirement ; (b) on obtient une population (c) la performance (la fitness) de chaque solution est évaluée ; (d) les solutions sont classées en fonction de leur fitness ; (f) les solutions les mieux classées sont sélectionnées pour générer une descendance par croisement puis mutation ; (e) la population parente est détruite ; le cycle recommence en (b).

l'ensemble de la génétique moderne. Cette théorie synthétique de l'évolution a été baptisée *néo-darwinisme* et correspond aux bases de l'approche en vigueur dans la communauté scientifique. Dans le langage génétique moderne, les molécules d'ADN, constituées d'une suite de nucléotides, sont le support du génotype – l'ensemble des gènes – dont l'expression est dénommée le phénotype. Bien qu'il n'existe pas de définition précise de la notion de gène qui soit universellement reconnue, on peut considérer qu'un gène est une séquence de nucléotides codant pour un caractère particulier, le plus souvent une protéine.

Simuler l'évolution sur un ordinateur afin de créer des machines plus « intelligentes » anime Turing dès 1948. Mais il faut attendre le milieu des années 1960 pour que se développent les premiers algorithmes appliquant les concepts du néo-darwinisme à l'informatique. Ces recherches ont mené à quatre grandes familles d'algorithmes évolutionnistes (Goldberg, 1989; Baeck et al., 1997) : les algorithmes génétiques (Holland, 1975), les stratégies d'évolution (Schwefel, 1993), la programmation génétique (Koza, 1992) et la programmation évolutionniste (L. J. Fogel, 1966). Deux nouvelles familles ont émergé plus récemment, les algorithmes à estimation de distribution (EDA, voir par exemple Larrañaga et Lozano (2002)) et les algorithmes multiobjectifs (voir (Deb, 2001)) sur lesquels nous nous attarderons dans la section suivante.

Toutes ces techniques reposent sur les mêmes principes de base (figure 2.1). Tout d'abord, elles travaillent sur une population, initialement aléatoire, d'individus représentant chacun une solution au problème traité. Ensuite, chaque

solution est représentée par un génotype et s'exprime sous la forme d'un phénotype. Enfin, il est nécessaire de mettre au point une fonction d'adaptation, la *fitness*, qui permet de distinguer les phénotypes les plus performants. Ces derniers ont une probabilité plus importante de léguer à leur descendance leur génotype. Les règles régissant cette transmission de gènes sont décrites sous la forme d'opérateurs génétiques. Trois opérateurs sont requis :

- l'opérateur de sélection qui décrit la manière dont les candidats sont choisis pour la reproduction ;
- l'opérateur de croisement (*cross-over*) ou de recombinaison qui correspond à la méthode utilisée pour mélanger les génotypes des parents ;
- l'opérateur de mutation qui décrit la fréquence et la nature des changements affectant le génotype lors de la transmission des gènes.

Il existe une multitude de versions de chaque opérateur, plus ou moins spécialisées pour le problème traité. Dans la suite, on se référera essentiellement à trois opérateurs de sélection (des détails pourront par exemple être trouvés dans (Deb, 2001) ou (Goldberg, 1989)) :

- la roulette biaisée (*roulette wheel*), qui permet de sélectionner les individus proportionnellement à leur fitness ;
- la roulette biaisée sur le rang, sélectionnant les individus en fonction de leur classement, éliminant ainsi les effets dus à des différences de fitness trop grandes dans la roulette biaisée classique ;
- le tournoi, dans lequel on tire aléatoirement $n$ individus et choisit celui qui a la meilleure fitness.

Les autres opérateurs, croisement et mutation, dépendent du codage utilisé. Dans le cas des algorithmes génétiques, les solutions potentielles sont représentées par des chaînes binaires (par exemple `0001`). L'opérateur principal est alors le croisement auquel s'ajoute la mutation pour ajouter de la diversité à la population. Dans le cadre de la programmation génétique, une solution est encodée sous la forme d'un programme. Le programme est exécuté afin de permettre le calcul de la fitness. Le génotype est vu comme un arbre où chaque nœud représente une sous-fonction, une structure de contrôle, un opérateur, ou un terminal. L'opérateur principal est alors là aussi le croisement, implémenté sous la forme d'échanges de sous-arbres. À l'inverse, la programmation évolutionniste et les stratégies d'évolution utilisent essentiellement l'opérateur de mutation. Le premier cas correspond à l'évolution de programmes, le second à celle d'une liste de paramètres. Actuellement, la distinction entre ces quatre approches est de plus en plus floue, le génotype étant souvent constitué d'un mélange de structures complexes (arbres, graphes, listes de paramètres, ...). La différences entre ces quatre approches est donc essentiellement d'ordre historique et ils sont maintenant considérés comme des instances particulières des *algorithmes évolutionnistes*(de Jong, 2002). Du point de vue de la théorie de l'optimisation, ces algorithmes constituent des métaheuristiques d'optimisation globale, au même titre que le recuit simulé (Kirkpatrick et al., 1983) et la recherche Tabou (Glover et al., 1997). Comme les autres méta-heuristiques,

leur principal intérêt repose dans le peu de contraintes imposées à la fonction à optimiser. Notamment, cette dernière n'a pas besoin d'être dérivable, contrairement aux méthodes à base de gradient. De plus, l'utilisation d'une population est une mesure efficace pour éviter certains minima locaux et celle-ci peut être efficacement exploitée pour optimiser plusieurs objectifs simultanément.

### 2.1.2 *Optimisation multiobjectif et notion de compromis*

De nombreux problèmes concrets requièrent l'optimisation de plusieurs objectifs simultanément. Par exemple, pour optimiser les paramètres d'un robot marcheur, on peut souhaiter minimiser la dépense énergétique tout en maximisant sa vitesse de déplacement. Dans leur forme la plus générale, les problèmes d'optimisation multiobjectif sont décrits de la manière suivante (Deb, 2001) :

$$\left.\begin{array}{rll} \text{Minimiser / maximiser :} & f_m(\mathbf{x}) & m = 1,2,\cdots,M; \\ \text{Sous les contraintes :} & g_j(\mathbf{x}) \geq 0 & j = 1,2,\cdots,J; \\ & h_k = 0 & k = 1,2,\cdots,K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)} & i = 1,2,\cdots,n. \end{array}\right\} \quad (2.1)$$

Une solution $\mathbf{x}$ est typiquement un vecteur de $n$ variables $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$, potentiellement bornées par $x_i^{(L)}$ et $x_i^{(U)}$. Néanmoins, on cherche parfois à optimiser d'autres structures, par exemple des graphes ou des permutations. Dans ce cas, $\mathbf{x}$ sera la représentation d'une solution, contrainte par l'espace atteignable associé. La programmation linéaire peut être efficacement utilisée si les objectifs et les contraintes sont linéaires. Nous nous intéressons donc ici essentiellement aux problèmes où au moins l'une des fonctions ou contraintes est non-linéaire.

La notion de *compromis* est fondamentale lors de l'optimisation simultanée de plusieurs objectifs. Imaginons quelqu'un souhaitant acheter une voiture et pour lequel deux critères sont importants, le prix et le confort. Une voiture plus confortable étant plus chère qu'une moins confortable, ces deux critères ne sont pas compatibles : il n'existe pas de solution maximisant simultanément les deux objectifs. Il existe cependant des modèles de voiture qui peuvent être éliminés de la sélection par l'acheteur, ceux pour lesquels il existe un autre modèle à la fois plus confortable et moins cher. Le choix devra donc se faire entre un ensemble de modèles qui sont tous de « bons » compromis, c'est-à-dire pour lesquels il n'existe pas de solution meilleure sur tous les objectifs. Ce concept est élégamment capturé par la notion de *domination de Pareto* (figure 2.2) et de *front de Pareto* (*Pareto-optimal set*).

**Définition 1** *Une solution $x^{(1)}$ est dite dominante par rapport à la solution $x^{(2)}$ si les conditions 1 et 2 sont vraies :*

1. *la solution $x^{(1)}$ n'est pas plus mauvaise que $x^{(2)}$ sur tous les objectifs ;*
2. *la solution $x^{(1)}$ est strictement meilleure que $x^{(2)}$ pour au moins un objectif.*

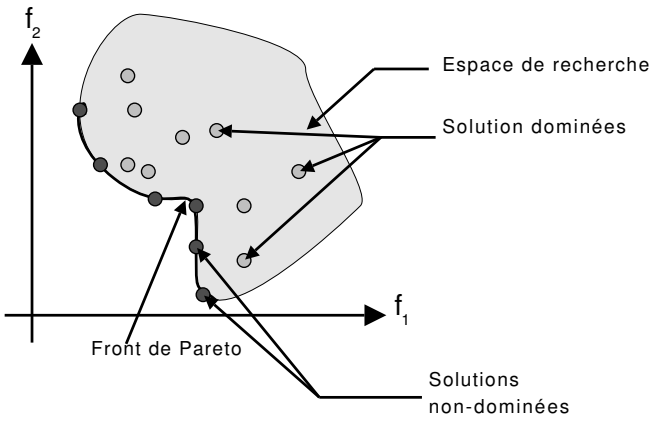



FIG. 2.2.: Parmi les différentes solutions possibles à un problème d'optimisation donné, pour lequel les deux objectifs f1 et f2 doivent être minimisés, les solutions non-dominées sont représentées en utilisant un disque sombre et celles dominées par un disque clair. Aucune solution n'est meilleure qu'une solution foncée à la fois sur f1 et f2.

**Définition 2** *Le front de Pareto est l'ensemble des solutions non dominées de l'espace de recherche.*

Le but d'un algorithme multiobjectif est donc de fournir un ensemble de compromis optimaux à l'utilisateur et non une unique solution comme c'est le cas en optimisation classique. Le compromis final est choisi, dans un deuxième temps, par un autre algorithme ou par l'utilisateur. Afin de pouvoir faire un choix correctement informé, l'utilisateur a besoin de connaître l'ensemble du front de Pareto. Une procédure d'optimisation multiobjectif a donc deux grands objectifs :

- trouver un ensemble de solutions non-dominées ;
- répartir ces solutions de manière à couvrir le front de Pareto le mieux possible.

Les méthodes classiques agrègent les différents objectifs en un seul, l'approche la plus courante étant d'effectuer une somme pondérée des objectifs. Dans ce cas, les poids fixent le compromis et on pourrait donc penser que lancer successivement l'algorithme avec des poids différents permettrait d'obtenir l'ensemble du front de Pareto. Ce n'est malheureusement pas le cas lorsque l'espace de recherche n'est pas convexe (figure 2.3) et il est donc nécessaire d'employer des méthodes spécifiques. Les algorithmes évolutionnistes multiobjectifs sont particulièrement efficaces dans ce contexte, notamment en raison de leur capacité naturelle à manipuler un ensemble de solutions.

$$\left.\begin{array}{rll} \text{Minimiser :} & f_1(x) = x^2 & \\ & f_2(x) = \frac{1}{x^5+1} & \\ \text{Sous la contrainte :} & -1 \leq x \leq 1 & \end{array}\right\}$$

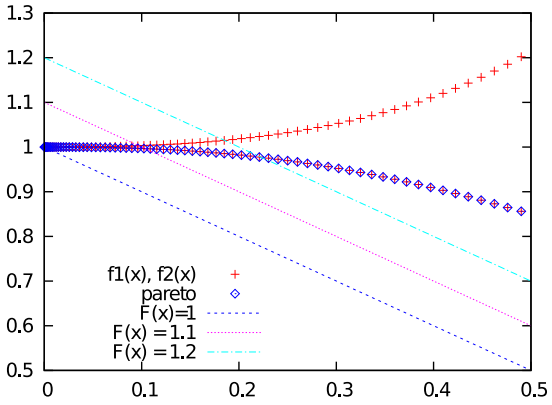



FIG. 2.3.: Exemple de problème d'optimisation multiobjectif où l'approche de la somme pondérée ne peut pas fonctionner. L'ensemble des points avec la même somme pondérée $F(x) = \alpha f_1(x) + \beta f_2(x)$ forme une droite dont le coefficient directeur est fixé par les poids $\alpha$ et $\beta$. Quels que soient les poids choisis, l'optimum de $F(x)$ restera $x = 0$. Le reste du front de Pareto, entouré par des carrés sur le graphe, ne pourra donc jamais être trouvé avec cette approche.

### 2.1.3 *Algorithmes de première génération*

La première implémentation d'un algorithme évolutionniste multiobjectif (MOEA) a été suggérée par Shaffer (1985) sous la forme de VEGA (Vector Evaluated Genetic Algorithm). Bien que mettant en valeur la nécessité d'employer des méthodes particulières pour l'optimisation multiobjectif, cette approche s'est révélée peu efficace au delà de quelques générations. En décrivant VEGA dans son livre, Goldberg (1989) suggéra l'utilisation du concept de domination pour sélectionner les individus pour la reproduction. Afin de préserver la diversité sur le front de Pareto, il proposa l'ajout de stratégies de niches. Ces considérations ont mené à une première génération d'algorithmes, différant uniquement dans leur manière d'affecter une fitness (et donc de classer) les individus : MOGA (Multi-Objective Genetic Algorithm), NSGA (Non Dominated Sorting Genetic Algorithm) et NPGA (Niched Pareto Genetic Algorithm).

MOGA (Fonseca et Fleming, 1993) assigne un rang $r_x$ à chaque solution $x$ correspondant au nombre de solutions qui dominent $x$ plus 1. Les solutions non-dominées sont alors de rang 1. Afin de maintenir la diversité sur le front de Pareto, MOGA exploite le principe des niches (Goldberg et Richardson, 1987),

qui consiste à dégrader la fitness des solutions similaires. Cette similarité se base sur une distance euclidienne dans l'espace des objectifs entre chaque solution, en ne prenant en compte que les solutions proches. Une roulette biaisée est utilisée pour sélectionner les individus. NSGA (Srinivas et Deb, 1994) commence par diviser la population en strates en fonction de la relation de domination : la première strate est constituée par les solutions non dominées, la seconde par celles qui ne sont dominées que par celles de la première strate, etc. La diversité est maintenue en utilisant des niches dans l'espace de décision (et non pas dans l'espace des objectifs). NPGA (Horn et al., 1994) utilise une sélection par tournoi binaire prenant en compte des niches dans l'espace des objectifs.

### 2.1.4 *Algorithmes élitistes*

L'élitisme est une méthode souvent utilisée dans les algorithmes évolutionnistes classiques qui consiste simplement à garder, de générations en générations, une partie des meilleurs individus dans la population, par exemple les 10% meilleurs. Cette approche permet de garantir que la fitness du meilleur individu ne décroît jamais et augmente la probabilité de créer une bonne descendance. De plus, l'élitisme est requis pour les preuves théoriques de convergence des algorithmes génétiques (Rudolph, 1996). Son introduction dans les algorithmes multiobjectifs a mené à des algorithmes de deuxième génération.

Le premier MOEA élitiste semble être SPEA (Strength Pareto Evolutionary Algorithm, Zitzler et Thiele (1999)), dans lequel une population externe $\bar{P}$, de taille bornée, contient l'élite. À chaque génération, le front de Pareto est comparé avec l'élite existante et l'ensemble des solutions non-dominées est préservé dans $\bar{P}$. Lorsque l'élite grossit, il est nécessaire de choisir quelles solutions éliminer. Pour cela, les auteurs utilisent un algorithme de classification (*clustering*) en choisissant de ne conserver que les représentants de chaque classe. Pour chaque individu de $\bar{P}$, une fitness (appelée *force*) proportionelle au nombre d'individus de la population $P$ qu'il domine est ensuite calculée. La force d'un individu de la population principale est, quant à elle, égale à la somme des forces des individus de $\bar{P}$ qui le dominent plus 1. Une sélection par tournoi binaire est utilisée, favorisant les individus avec la force la plus faible. SPEA a depuis été amélioré pour donner SPEA-2 (Zitzler et al., 2001). Ce nouvel algorithme utilise un schéma d'assignation de fitness plus fin, une méthode d'estimation de la densité et une gestion de la réduction de $\bar{P}$ améliorée.

NSGA-II (Deb et al., 2000b) est un autre algorithme élitiste, basé sur le classement des individus en strates (aussi appelées fronts) selon la relation de domination. Nous le décrivons ici en détail car nous nous en servirons dans la suite de ce manuscrit. À chaque itération, une population $Q_t$ est créée à partir de la population parente $P_t$, chacune de taille $N$. Les deux populations sont alors combinées pour former une population $R_t$, de taille $2N$. Un classement en strates de Pareto (*non dominated sorting*) est ensuite calculé entre les membres

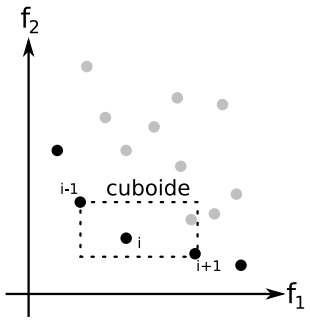



FIG. 2.4.: Pour une solution $i$ du front $F$ (formé par les ronds noirs), la distance de peuplement (*crowding distance*) d'une solution $i$, $d_i$, utilisée dans NSGA-II correspond au périmètre du cuboïde formé par ses deux voisins. D'après (Deb, 2001).

de $R_t$, à partir duquel on construit la nouvelle population $P_{t+1}$. Les fronts sont successivement ajoutés à $P_{t+1}$ en commençant par le front de Pareto, puis le deuxième front, etc. Comme $R_t$ est beaucoup plus grande que $P_{t+1}$, tous les fronts ne pourront pas être ajoutés. Si un front ne peut pas être ajouté en entier, une stratégie de niche, appelée peuplement (*crowding*), est employée pour sélectionner les individus à ajouter. Enfin, un tournoi prenant en compte le peuplement (*crowded tournament*) est utilisé pour générer $Q_{t+1}$ à partir de $P_{t+1}$. Voici la procédure détaillée, telle que la décrit Deb (2001) :

1. Combiner les populations parentes $P_t$ et descendantes $Q_t$ et créer $R_t \leftarrow P_t \cup Q_t$. Effectuer un classement en strates de Pareto et identifier les différents fronts $F_i, i = 1, 2, \cdots$.
2. $P_{t+1} \leftarrow \emptyset$, $i \leftarrow 1$. Faire $P_{t+1} \leftarrow P_{t+1} \cup F_i$ et $i \leftarrow i+1$ tant que $|P_{t+1}| + |F_i| < N$.
3. Appeler la procédure de classement par peuplement (décrite dans la suite) et inclure les $(N - |P_{t+1}|)$ solutions les mieux réparties de $F_i$ dans $P_{t+1}$.
4. Créer $Q_{t+1}$ à partir de $P_{t+1}$ en utilisant le tournoi prenant en compte le peuplement (voir ci-après).

Pour obtenir une estimation de la densité des solutions au voisinage d'une solution particulière $i$ sur un front $F$, NSGA-II calcule la distance moyenne des deux solutions de chaque côté de $i$ selon chaque objectif. Cette quantité $d_i$ sert d'estimation du périmètre du cuboïde formé par les plus proches voisins autour de $x$ (figure 2.4). Cette procédure peut se calculer en trois étapes, pour un front $F$ et une *distance de peuplement* $d_i$ :

1. Pour chaque solution $i$ dans $F$, $d_i \leftarrow 0$. On note $l = |F|$.
2. Pour chaque objectif $m = 1, 2, \cdots, M$, classer les solutions en ordre décroissant selon $f_m$ et créer un vecteur d'indices classés $I^m = sort(f_m, >)$, où $I^m_j$ correspond au rang de la $j$-ième solution de $F$ classée selon $f_m$.

3. $d_{I_1^m} = d_{I_l^m} = \infty$ et pour toutes les autres solutions $j = 2$ à $(l-1)$ :

$$d_{I_j^m} \leftarrow d_{I_j^m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{\max} - f_m^{\min}}$$

L'opérateur de sélection, le tournoi avec peuplement (*crowded tournament*), sélectionne en priorité les solutions non dominées puis, si aucune des solutions tirées aléatoirement ne domine l'autre, en fonction de la distance de peuplement. Plus formellement, une solution $i$ gagne un tournoi contre une autre solution $j$ si une des conditions suivantes est vraie :

1. si la solution $i$ a un meilleur rang ($r_i < r_j$) ;
2. si elles ont le même rang et que la solution $i$ a une meilleure distance de peuplement que la solution $j$ ($r_i = r_j$ et $d_i > d_j$).

### 2.1.5 $\varepsilon$-*domination*

Peu de preuves théoriques sur la convergence des MOEA existent, bien qu'ils donnent le plus souvent d'excellents résultats en pratique (Coello et Lamont, 2004). Rudolph et Agapie (2000) suggèrent une série d'algorithmes qui garantissent la convergence mais qui ne prennent pas en compte la répartition des solutions sur le front de Pareto ni la complexité en temps pour converger vers le « véritable » front de Pareto. L'introduction du concept d'$\varepsilon$-domination (Laumanns et al., 2002) a permis de proposer de nouveaux algorithmes combinant des garanties de convergence et de répartition des solutions. L'$\varepsilon$-domination correspond à la relation de domination de Pareto à un $\varepsilon$ près, définie par l'utilisateur de l'algorithme :

**Définition 3** *Soient $m$ le nombre d'objectifs, $f$ et $g \in \Re^{+^m}$ deux vecteurs. $f$ $\varepsilon$-domine $g$ pour un $\varepsilon > 0$ si et seulement si, pour tout $i \in \{1, \cdots, m\}$ :*

$$(1+\varepsilon) \cdot f_i \geq g_i$$

L'utilisation de l'$\varepsilon$-domination a mené à de nouveaux algorithmes, dont une variante de NSGA-II, $\varepsilon$-NSGA-II (Deb et al., 2005) et $\varepsilon$-MOEA (Deb et al., 2005). Ce dernier est un algorithme *steady-state* élitiste qui utilise une archive $E$ en plus de la population $P$. Lors de la phase de reproduction, un des parents est tiré au hasard dans $E$, l'autre est issu d'un test de domination classique entre deux solutions aléatoirement choisies dans $P$. Les solutions générées par croisement et mutation sont ensuite comparées à $P$ et $E$ pour leur possible inclusion. Pour l'inclusion dans $E$, un test d'$\varepsilon$-domination est effectué. Si la nouvelle solution domine des membres de l'archive, ces derniers sont éliminés et la nouvelle solution est ajoutée. Sinon, si elle n'est pas dominée et n'est dominée par aucun membre de l'archive, la solution la plus proche du centre de la « case » définie par $\varepsilon$ est retenue. Une solution remplace un individu de $P$ qu'elle domine ; inversement, si un membre de $P$ la domine, elle est refusée ; enfin, si ces deux tests échouent, elle remplace un membre aléatoire de $P$.

### 2.1.6 *Comparaisons*

D'autres MOEA élitistes ont été proposés, notamment PAES (Pareto Archived Evolution Strategy, Knowles et Corne (2000)) et son extension PESA (Corne et al., 2001). Les différents algorithmes ont été comparés à plusieurs reprises sur des fonctions artificielles (Zitzler et al., 2000; Deb, 2001; Deb et al., 2005; Kollat et Reed, 2005) et sur certains problèmes réels (Reed et al., 2007). Les algorithmes de deuxième génération se montrent bien plus performants que leurs homologues de la génération précédente. Parmi eux, NSGA-II et SPEA-2 se distinguent clairement. Bien que la différence entre ces deux méthodes ne soit pas significative sur des problèmes avec deux objectifs (Deb et al., 2000b), les solutions sont bien mieux réparties avec SPEA-2 sur les problèmes à trois objectifs (Zitzler et al., 2000). Néanmoins, les meilleures performances de SPEA-2 ont un coût computationnel fort en raison de la complexité de l'algorithme de classification ( en $O(N^3)$). Les études les plus récentes semblent montrer la supériorité des approches basées sur l'$\varepsilon$-domination. Kollat et Reed (2005) et Deb et al. (2005) concluent tous deux à l'avantage d'$\varepsilon$-MOEA sur NSGA-II et SPEA-2. Néanmoins, Reed et al. (2007) suggèrent que la variante $\varepsilon$-NSGA-II pourrait être plus performante que d'$\varepsilon$-MOEA.

## 2.2 ÉVOLUTION ET MODULARITÉ

### 2.2.1 *Ubiquité et définition de la modularité*

Les ingénieurs ont reconnu depuis longtemps la nécessité de commencer par concevoir des modules simples puis de les combiner dans des structures plus complexes. Les sous-parties des machines ainsi conçues peuvent ensuite être aisément réutilisées dans de nombreux contextes ; dans la même machine, pour réaliser des économies d'échelles, ou dans d'autres projets, pour réduire les coûts d'ingénierie. De plus, de telles machines sont souvent plus robustes que des machines non modulaires car certaines sous-parties peuvent être aisément remplacées. En informatique, la programmation objet (Meyer, 1997a), la programmation par composant (Szyperski, 1998) ou l'utilisation de tests unitaires (Zhu et al., 1997) sont des exemples remarquables de la notion de module. Le concept de modularité tient aussi une place prépondérante dans les sciences du vivant (Carroll, 2001; Schlosser et Wagner, 2004; Hartwell et al., 1999). L'importance de la conception des animaux à partir de sous-parties répétées a été reconnue depuis longtemps (Cope, 1871; Bateson, 1894; Gregory, 1935) ainsi que dans les plantes (Niklas, 1994), où la structure modulaire permet notamment la différentiation des rôles fonctionnels entre les structures reproductrices et les feuilles. La modularité des organismes vivants et de leur génome a été spectaculairement montrée par Halder et al. (1995) lorsqu'ils ont réussit à faire pousser des yeux fonctionnels sur les ailes, les pattes et les antennes d'une drosophile après modification d'une très petite partie de son gé-

nome. La modularité du vivant semble se manifester à de très nombreux niveaux comme la structure des gènes individuels (Yuh, 1996), des caractères morphologiques contribuant au phénotype (Cheverud, 1996), les réseaux régulateurs de gènes (Force et al., 2005; Bongard, 2002) ou la biologie moléculaire (Hartwell et al., 1999). En neuro-psychologie, de nombreuses études par double dissociation (Teuber, 1955), dans lesquelles des déficits sont reliés à des dommages cérébraux, suggèrent une organisation modulaire du cerveau animal (voir par exemple (Shallice, 1988)), bien que celle-ci reste controversée. L'imagerie fonctionnelle à résonance magnétique (functional magnetic resonance imagery, fMRI) apporte une méthode alternative pour affiner et parfois remettre en cause les observations effectuées avec des lésions du cerveau (Huettel et al., 2004; Anderson, 2007). Cette approche a par exemple récemment permis d'analyser et de questionner la modularité de l'itinéraire objet-vision (de Beeck et al., 2008).

L'ubiquité des concepts de modularité et de modules ne facilite pas la définition de quelque chose que l'on reconnaît pourtant, le plus souvent facilement, à savoir une organisation sous forme d'entités distinctes interagissant entre elles à différentes échelles. Dans cette thèse, nous utiliserons la définition de module suivante, similaire à celles utilisée dans de nombreux travaux (Bolker, 2000; Force et al., 2005; Callebaut et Rasskin-Gutman, 2005; Hartwell et al., 1999; Simon, 1962) :

**Définition 4** *Un module est un sous-ensemble d'un système contenant plusieurs entités fonctionnellement intégrées et largement indépendantes des entités constituant les autres modules.*

Dans le cadre de l'évolution de réseaux de neurones, les interactions entre neurones sont définies par la présence de connexions et par leurs poids synaptiques. Un ensemble de neurones fortement connectés entre eux mais peu reliés au reste du réseau aura donc des interactions limitées avec les autres sous-réseaux. Nous pouvons donc définir une *modularité structurelle* à partir de la topologie du réseau en définissant un module comme un sous-réseau dont le nombre de connexions extra-module est plus petit que le nombre équivalent dans un réseau aléatoire aux caractéristiques statistiques similaires (Newman, 2006a). Nous reviendrons par la suite sur cette définition afin de décrire une mesure de modularité. À ce stade, il est important de noter que la modularité structurelle ne correspond pas nécessairement à la *modularité fonctionnelle* : un sous-réseau faiblement connecté au reste du réseau n'effectue pas nécessairement une fonction indépendante. Pour constituer un module, un sous-réseau de neurones doit donc combiner une relative indépendance topologique *et* effectuer une sous-fonction identifiable indépendamment des autres modules.

### 2.2.2 *Modularité et évolvabilité*

Un système biologique est dit évolutif (*evolvable*) s'il peut acquérir des fonctions nouvelles grâce à des changements génétiques, fonctions qui aident l'organisme à survivre et à se reproduire. La capacité d'un organisme à être évolutif est appelée *evolvabilité* (*evolvability*, voir Wagner (2005)). Selon de nombreux auteurs (Simon, 1962; Lipson, 2004; de Jong et Thierens, 2004; Hornby, 2005; Wagner et Altenberg, 1996), elle serait augmentée par la modularité. Ainsi, analysant l'évolution des métazoaires, Carroll (2001) conclut : « un des plus importants aspects qui a facilité l'évolution de la complexité et de la diversité des plantes et des animaux est la modularité de leur construction à partir de composants réitérés et différenciés ». De manière théorique, le lien entre modularité et evolvabilité a été élégamment mis en valeur par Simon (1962) avec la parabole des deux horlogers :

> « Il était une fois deux horlogers, nommés Hora et Tempus, qui construisaient des montres d'excellente facture. [...] Les montres étaient constituées chacune d'environ mille pièces. Tempus construisait la sienne de telle sorte que s'il ne l'avait que partiellement assemblée et qu'il était interrompu, elle tombait immédiatement en morceaux et devait être ré-assemblée à partir de ses éléments. [...] Les montres que Hora faisaient n'étaient pas moins complexes que celles de Tempus. Mais il les avait conçues de telle sorte qu'il puisse assembler des sous-ensembles de dix éléments chacun. Dix de ces ensembles pouvaient ensuite être réunis dans un ensemble plus grand ; et un système de dix de ces derniers ensembles constituait la montre entière. Ainsi, quand Hora avait à déposer une montre partiellement assemblée [...], il ne perdait qu'une partie de son travail, et il construisait ses montres en une fraction du temps nécessaire à Tempus. »

Il est aisé de comprendre que plus la complexité des systèmes augmentera, plus l'approche de Hora – basée sur modularité – sera efficace. Simon conclut que le temps requis pour l'évolution d'une forme complexe à partir de formes plus simples dépend de manière critique du nombre et de la distribution des formes stables intermédiaires potentielles. De plus, la modularité réduit la quantité de couplages entre les changements internes et externes, permettant ainsi à l'évolution de réorganiser les entrées des modules sans changer leur comportement intrinsèque. Les modules peuvent alors être utilisés comme « pièces de constructions » pour de nouvelles créations ou permettre une adaptation plus rapide d'une structure existante. Dans la nature, cette idée est soutenue par des arguments théoriques suggérant que les protéines évoluent plus difficilement une fois qu'elles participent à beaucoup d'interactions différentes (Waxman et Peck, 1998).

L'exemple des horlogers met aussi l'accent sur le concept de *hiérarchie*, indissociable de celui de module. L'intégration, fonctionnelle ou structurelle,

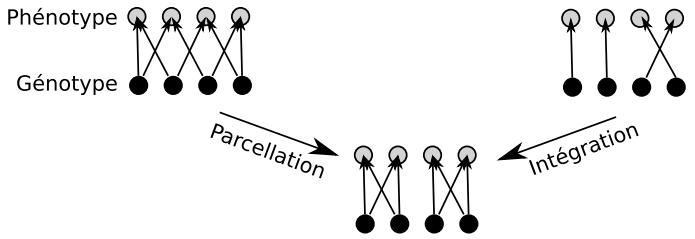



FIG. 2.5.: Une organisation modulaire peut émerger de deux façons : la parcellation (élimination des effets pléiotropiques extra-modules et le maintien et/ou l'augmentation des liens intra-modules) et l'intégration (création d'effets pléiotropiques entre des phénotypes initialement indépendants).

des composants d'un module n'est valable qu'à une certaine échelle ou à un certain niveau d'analyse ; ce qui est défini comme un module à l'échelle des molécules n'en est pas nécessairement un à l'échelle des organes. Le concept de *régularité* (ou répétition), défini comme la réutilisation du même module plusieurs fois, complète la hiérarchie et la modularité pour donner, selon certains auteurs, les trois ingrédients nécessaires à l'évolution de systèmes complexes (Lipson, 2004; de Jong et Thierens, 2004; Hornby, 2005). Pour illustrer ce propos, Hornby (2005) a tenté de mesurer l'influence de la modularité, de la hierarchie et de la régularité sur l'évolution de la structure de tables. Un codage basé sur la programmation génétique avec des fonctions automatiquement définies a été utilisé. Ses résultats montrent une corrélation positive importante entre l'augmentation de la fitness et l'augmentation des mesures de modularité, de régularité et de hiérarchie.

### 2.2.3 *Origine évolutionniste de la modularité*

Si l'omniprésence de la modularité et son intérêt pour l'évolvabilité semblent faire consensus, les conditions précises de son émergence sont encore mal comprises. Est-elle liée a des caractéristiques du génome ? Est-elle uniquement le résultat d'une pression sélective particulière ? Si oui, laquelle ? Cette question, intéressante pour comprendre l'évolution des systèmes vivants, trouve un écho important pour l'évolution d'artefacts : comment peut-on obtenir par évolution artificielle des systèmes modulaires ? Ces interrogations sont d'autant plus pertinentes que la modularité semble être absente dans la plupart des résultats des expériences d'évolution artificielle. Lors de l'évolution de réseaux de neurones, par exemple, les réseaux non modulaires sont généralement plus performants et obtenus plus facilement par évolution que leurs homologues modulaires (Bowers et Bullinaria, 2005).

En biologie, la modularité peut être étudiée au niveau du génotype, du phénotype et des liens unissant génotype et phénotype. On peut considérer le génotype comme un ensemble de gènes ayant chacun une fonction, soit la production d'une protéine ou un changement physico-chimique dans l'état de la cellule. Les gènes forment alors un *réseau régulateur de gènes* (*gene regulatory network*, GRN) qui peut être représenté par un graphe orienté dont les sommets correspondent aux gènes et les arcs aux dépendances. Tous les gènes n'influençant pas tous les gènes, des modules peuvent être distingués dans ce graphe. Si ces modules et leurs propriétés ne sont pas tous connus, leur simple observation permet de conclure que le génotype est modulaire. De la même manière, l'analyse de l'anatomie des êtres vivants montre clairement une organisation modulaire du phénotype. Cependant, étudier la modularité du phénotype et du génotype indépendamment a peu de sens pour comprendre les liens unissant modularité et évolvabilité. Il est plus pertinent de se demander si la présence de modules permet de générer des phénotypes utiles à l'organisme en modifiant le génotype.

Pour répondre à une telle question, il faut étudier les processus permettant d'associer un ensemble de gènes à un ensemble de traits phénotypiques, regroupés sous le nom de *correspondance génotype-phénotype* (*genotype-phenotype map*). Cette correspondance peut être vue comme modulaire si elle peut être décomposée en plusieurs correspondances indépendantes de plus petites dimensions (Altenberg, 2005). Il semble intuitif que l'évolution puisse bénéficier d'une telle modularité car si les changements génétiques tendent à n'influencer que peu de traits phénotypiques, alors le génome peut répondre à la sélection pour uniquement ces traits, indépendamment du reste du phénotype. Cette modularité améliorerait alors la capacité du système à générer des variantes adaptées.

L'étude de la correspondance génotype-phénotype se ramène souvent à l'analyse de la *pléiotropie*, c'est-à-dire les situations où un unique gène influence plusieurs caractères phénotypiques. Les effets pléiotropiques réduisent généralement la modularité en augmentant le nombre de gènes dont dépend un caractère mais ils peuvent aussi augmenter la régularité en permettant de coder plusieurs traits phénotypiques avec les mêmes gènes. Dans ce contexte, un module d'évolution est « un ensemble de caractéristiques phénotypiques qui sont hautement intégrées par les effets phénotypiques des gènes sous-jacents et relativement isolées des autres ensembles par une pauvreté d'effets pléiotropiques » (Altenberg, 2005).

La modularité dans la correspondance génotype-phénotype ne peut émerger que de deux manières, par parcellation ou par intégration (Wagner (1996), figure 2.5). La parcellation consiste en l'élimination progressive des effets pléiotropiques parmi les caractères qui appartiennent à différents complexes fonctionnels. À l'opposé, l'intégration peut être vue comme une factorisation où différents caractères fonctionnels initialement indépendants sont regroupés dans un même gène. La parcellation et l'intégration ne sont pas mutuellement

exclusives et il est surtout intéressant de comparer leur fréquence respective d'apparition.

De nombreux mécanismes évolutionnistes pouvant mener à des systèmes modulaires ont été proposés mais aucun ne fait l'unanimité à ce jour. Wagner et al. (2005) en font une analyse complète dont nous ne présentons ici que les grandes lignes. En outre, il est probable que plusieurs mécanismes œuvrent simultanément ou que certaines des hypothèses envisagées soient deux faces d'un même phénomène.

LA SÉLECTION DIRECTE POUR L'ÉVOLVABILITÉ. Puisque la modularité semble augmenter l'évolvabilité et que celle-ci a besoin d'être importante pour permettre l'émergence de systèmes complexes, il semble naturel que les systèmes les plus évolutifs (*evolvable*) aient été sélectionnés par l'évolution, favorisant ainsi la modularité. Pour approfondir cette hypothèse, il faut d'abord se demander si, dans un contexte plus large, la sélection pour l'évolvabilité peut être un facteur important dans l'évolution de la correspondance génotype-phénotype. Cette question n'est pas résolue actuellement (Wagner et al., 2005). Il a été montré par des simulations numériques que les effets pléiotropiques réduisent l'évolvabilité (Baatz et Wagner, 1997). Leur réduction devrait donc être sélectionnée. Mezey et al. (2000) ont étudié cette hypothèse en modélisant l'évolution des effets pléiotropiques et ont conclu que l'allèle supprimant les effets pléiotropique est sélectionné. Cependant, la fraction de l'avantage sélectif causé par la sélection pour l'évolvabilité était bien inférieure à 10%, les 90% restant étant dus à des avantages sélectifs directs. Dans un autre article, Turney (2000) arrive à des résultats similaires : l'évolvabilité augmente pendant les expériences mais elle est une conséquence d'une sélection pour d'autres avantages sélectifs. Wagner et al. (2005) concluent que la modularité n'est probablement pas le résultat d'une sélection directe pour l'évolvabilité.

LA SÉLECTION SUR LES EFFETS PLÉIOTROPIQUES. Une autre origine possible est la combinaison d'une force directionnelle importante, tendant à adapter l'organisme à son environnement, et une force stabilisatrice, tendant à éliminer l'effet des mutations (Wagner, 1996). Les fluctuations climatiques pourrait être une cause probable des épisodes de sélection directionnelle. Si l'on suppose que ces fluctuations peuvent prendre une forme variée, elles pourraient requérir l'adaptation de caractères à chaque fois différents comme la taille du bec, la forme des ailes, etc. À chaque fois, la sélection d'un des modules serait accompagnée par une sélection stabilisatrice pour les autres. Chaque épisode aurait donc pour effet la suppression des effets pléiotropiques entre les modules. Avec le temps, on peut donc s'attendre à observer une parcellation dans la structure de la correspondance génotype-phénotype.

À l'aide d'un modèle simplifié, Mezey et al. (2000) concluent que l'alternance des pressions de sélection ne mène pas à la séparation des gènes en deux ensembles séparés mais Lipson et al. (2002) arrivent à la conclusion in-

verse avec un autre modèle. Cette hypothèse a récemment été testée sur l'évolution de réseaux de neurones et de fonctions logiques (Kashtan et Alon, 2005). En utilisant un algorithme évolutionniste simple et un codage direct, Kashtan et Alon ont pu montrer que si la fitness alternait régulièrement (typiquement toutes les vingt générations) entre deux objectifs différents mais contenant des sous-fonctions communes, des modules calculant ces sous-fonctions émergeaient spontanément[1].

LA SELECTION CONSTRUCTIONNELLE. Altenberg (1994) se base sur la supposition que les gènes avec moins d'effets pléiotropiques ont une plus grande probabilité d'être copiés dans le génome pour construire un modèle prédisant la baisse de la pléiotropie moyenne dans le temps. Le principal problème de ce modèle est la supposition que le degré de pléiotropie est héritable parmi les copies des gènes, en particulier si ces gènes acquièrent de nouvelles fonctions. Il existe des preuves que la pléiotropie est plus faible parmi les copies de gènes mais d'autres phénomènes semblent mieux les expliquer (Force et al., 1999).

LA STABILITÉ PHÉNOTYPIQUE. Des résultats concernant la structure des protéines (Bornberg-Bauer et Chan, 1999) et l'évolution de la structure secondaire de l'ARN (Ancel et Fontana, 2000) suggèrent que la modularité pourrait constituer un effet de bord de l'évolution de la robustesse du phénotype contre les perturbations environnementales.

LA FACILITATION DE L'ADAPTATION PHYSIOLOGIQUE. En utilisant un algorithme évolutionniste pour faire évoluer l'architecture d'un réseau de neurones pour la tâche « where and what », Di Ferdinando et al. (2001) ont noté que les performances de l'algorithme d'apprentissage supervisé utilisé dépendaient beaucoup de l'architecture et que les topologies les plus performantes étaient modulaires. Ces résultats suggèrent un lien fort entre la modularité des réseaux de neurones et les phénotypes avec une haute fitness grâce à la plasticité.

LES CONSÉQUENCES DE LA "FRUSTRATION". En travaillant sur les mathématiques des interactions entre les gènes, Rice (2000) a pu montrer que si plus de trois caractères ont des interactions de fitness antagonistes, c'est-à-dire si l'augmentation de la fitness pour l'un baisse celle de l'autre, la seule solution stable est que ces caractères évoluent vers une indépendance variationnelle.

LES CONTRAINTES FONCTIONNELLES ET STRUCTURELLES. Une dernière hypothèse récente propose que la modularité pourrait émerger de la multiplicité des contraintes fonctionnelles et structurelles (Pan et Sinha, 2007) ; par

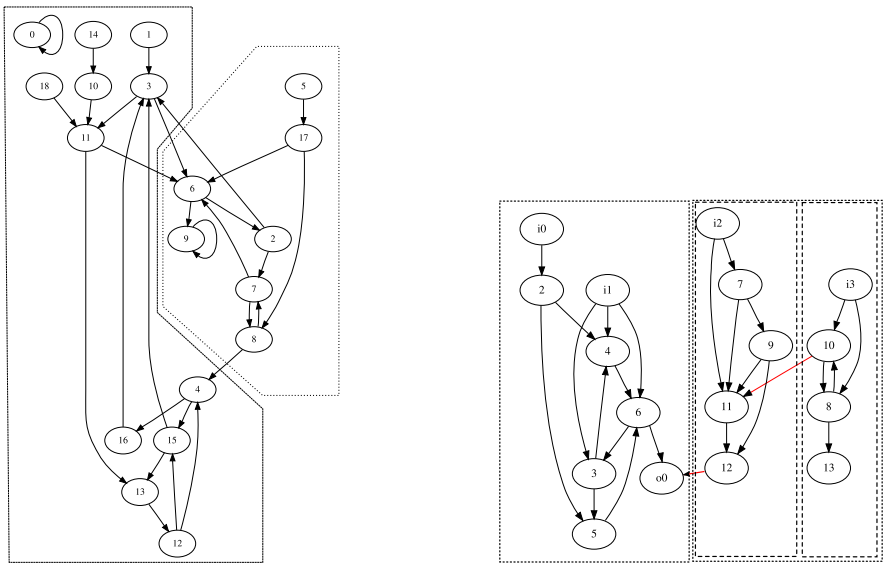
1 Nous n'avons malheureusement pas réussi à reproduire ces expériences dans un contexte très similaire.

exemple minimiser la longueur moyenne des chemins et le nombre total de connexions tout en maximisant la robustesse contre les perturbations dans l'activité des neurones.

### 2.2.4 *Mesure de la modularité*

Les études computationnelles de la modularité nécessitent la mesure du « niveau de modularité » d'un système ainsi que l'identification des modules. Si dans beaucoup de modèles de biologie théorique cités dans la section précédente des méthodes très simples peuvent être utilisées, l'analyse de la modularité des réseaux de neurones et des réseaux régulateurs de gènes requiert l'identification de modules dans des graphes. L'intérêt de la détection de ces structures couvre un très large spectre, de l'analyse des réseaux sociaux au réseaux de communication ou aux réseaux métaboliques ; elle a donc suscité de nombreuses recherches. Les interactions entre des composants ou la structure d'un système sont représentées sous la forme d'un graphe, orienté ou non, dans lequel on cherche à détecter des groupes de sommets densément connectés entre eux mais faiblement connectés aux sommets des autres groupes. Deux grandes voies de recherche semblent avoir été suivies pour découvrir de tels groupes : le partitionnement de graphe (Fjallstrom, 1998), issu de l'informatique, et la détection de communautés, initialement développée en sociologie mais aussi de plus en plus utilisée en biologie (White et al., 1976; Wasserman et Faust, 1994; Girvan et Newman, 2002). Bien que partageant un but très similaire, ces deux approches diffèrent selon deux grands axes. Tout d'abord, le nombre et la taille approximative des groupes sont généralement connus en partitionnement de graphes. Ensuite, le but de cette approche est le plus souvent de trouver le meilleur partitionnement, même si aucune bonne division existe. La détection de communautés est donc plus proche de ce qui nous intéresse pour l'évolution de réseaux de neurones : nous ne connaissons pas à l'avance le nombre de modules que l'on cherche et nous sommes très intéressés par l'estimation de la qualité d'une division car elle permet d'estimer à quel point les propriétés de modularité sont présentes.

Selon Newman (2006b), « une bonne division d'un réseau en communautés n'est pas simplement un découpage dans lequel il y a peu de lien entre les communautés, c'est une division dans laquelle il y a *moins de liens qu'attendu* ». Cela l'a amené à définir la quantité de modularité comme le nombre d'arcs dans des groupes auquel on soustrait le nombre d'arcs attendus dans un réseau équivalent avec des arrêtes placées aléatoirement. Cette quantité peut être négative ou positive, le dernier cas indiquant la possibilité de l'existence d'une structure en communautés. Cette définition de modularité précise celle que nous avons adoptée et décrite précédemment (définition 4) en transposant la notion d'« unités intégrées et largement indépendante » en une définition portant sur la connectivité des nœuds du graphe d'interaction. Elle a été utilisée pour analyser l'évolution de la modularité dans des expériences d'évolu-




(a) (b)

FIG. 2.6.: Deux exemples de graphes orientés aux caractéristiques statistiques similaires (nombre d'arcs et nombre de nœuds identiques) mais à la mesure de modularité différente. (a) Un réseau de faible modularité ($Q = 0.116343$), divisé en deux modules (b) Un réseau de forte modularité ($Q = 0.546713$), divisé en deux modules principaux dont l'un est sub-divisé.

tion de réseaux de neurones modulaires par Kashtan et Alon (2005) et a donné des résultats plus satisfaisant que les autres méthodes dans de nombreuses situations (Danon et al., 2005).

Plus formellement, pour un graphe contenant $n$ sommets décrit par une matrice de connexité $A_{ij}$ et une division donnée en deux groupes, on définit le vecteur $\mathbf{s}$ tel que $\mathbf{s}_i = 1$ si le sommet $i$ appartient au groupe 1 et $\mathbf{s}_i = -1$ s'il appartient au groupe 2. Le nombre *attendu* d'arcs entre les sommets $i$ et $j$ si les arrêtes sont placées au hasard est $k_i k_j / 2m$, où $k_i$ et $k_j$ sont les degrés des sommets $i$ et $j$ et $m$ est le nombre total d'arcs du graphe. La quantité de modularité $Q$ peut alors être écrite de la façon suivante :

$$Q = \frac{1}{4m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \mathbf{s}_i \mathbf{s}_j \tag{2.2}$$

Diviser un réseau en modules revient alors à trouver la partition qui maximise $Q$. Cette optimisation difficile peut être effectuée efficacement en se basant sur la décomposition spectrale d'une matrice de modularité, aboutissant à un algorithme non exact mais donnant de bons résultats en $O(n^2)$(Newman, 2006b,a). Pour diviser en plus de deux parties, il est possible, en modifiant légèrement la formule 2.2, de diviser récursivement chaque module en deux parties jusqu'à

la mesure de modularité ne s'améliore plus. On obtient alors une décomposition modulaire hiérarchique.

La figure 2.6 est un exemple de l'utilisation de la méthode de Newman (2006b) pour trouver les modules structurels d'un graphe et estimer sa modularité.

## 2.3 ÉVOLUTION DE RÉSEAUX DE NEURONES

### 2.3.1 *Introduction*

Un réseau de neurones est un graphe orienté pondéré dont les nœuds sont des unités de calculs : des neurones. De nombreux types de neurones peuvent être utilisés, correspondant chacun à différentes abstractions du neurone biologique. Dans le modèle proposé par McCulloch et Pitts (1943), la sortie $y_i$ d'un neurone est une fonction $\varphi(x)$ de la somme des signaux entrants $x_j$ pondérés par des poids synaptiques $w_{ij}$ :

$$y_i = \varphi\left(\sum_j w_{ij} x_j\right)$$

On utilise généralement pour la fonction $\varphi(x)$ une fonction de seuil dérivable, la plus courante étant la sigmoïde, paramétrée par une pente $k$ et un biais $b$ :

$$\varphi(x) = \frac{1}{1+\exp(b-kx)}$$

.

Si on organise de tels neurones sous forme de réseau à couches sans cycles (des réseaux non-récurrents ou *feed-forward*), appelés perceptrons multicouches, on peut prouver qu'ils permettent d'approcher avec une précision arbitrairement choisie n'importe quelle fonction continue à condition d'utiliser suffisamment de neurones. Ces capacités sont une des raisons de leur succès en apprentissage supervisé où ils sont généralement combinés à la rétro-propagation du gradient (*back-propagation*). Une bonne introduction aux réseaux de neurones dans le cadre de l'apprentissage supervisé peut être trouvée dans (Haykin, 1998) et dans (Bishop, 1995). Les intégrateurs à fuite (*leaky-integrators*), dans lesquels la sortie des neurones est calculée en utilisant une équation différentielle, sont un autre modèle de neurones, assez souvent utilisé en robotique et en neurosciences (e.g. Kodjabachian et al. (1999), Nolfi et Floreano (2001), Ijspeert et Kodjabachian (1999), Girard et al. (2003)) en raison de leur capacité d'approximateur temporel universel. Les neurones à décharge (*spiking neurons*, voir par exemple (Gerstner et Kistler, 2002)) forment une troisième famille de modèles, plus proche de la biologie, dans lesquels chaque potentiel d'action est traité séparément. La génération de réseaux de neurones par évolution est indépendante du type de neurones utilisé. Les méthodes décrites par la suite s'appliquent donc de la même manière à chacun de ces différents types.

Les algorithmes évolutionnistes peuvent être utilisés avec des réseaux de neurones à trois niveaux différents : le choix des poids synaptiques (ou, plus généralement, des paramètres), la mise au point d'une topologie adaptée et la sélection des règles d'apprentissage. Dans cette thèse, nous avons choisi de nous intéresser en priorité aux méthodes permettant l'évolution des poids synaptiques et de la topologie simultanément, sans restriction sur les topologies atteignables. Nous ne nous attarderons donc pas sur les approches restreintes aux perceptrons multicouches ni sur celles mettant en jeu une phase d'apprentissage. Une large revue des autres méthodes pourra être trouvée dans (Yao, 1999). Une grande partie des méthodes décrites dans cette section a été appliquée à la conception de contrôleurs pour des robots, sujet que nous aborderons dans la section 2.4.

L'essentiel des travaux portant sur l'évolution de réseaux de neurones se concentre sur la définition d'un codage et d'opérateurs génétiques permettant une exploration efficace des topologies et paramètres. Les codages proposés dans la littérature se divisent en deux grandes catégories, les codages directs et indirects (Kodjabachian et Meyer, 1995; Floreano et al., 2008). Dans le premier cas, le plus simple, il existe une bijection entre le génotype et le phénotype, chaque neurone et chaque connexion apparaissant indépendamment dans le génome. Un codage indirect, parfois appelé aussi codage dévelopemental, est une représentation plus complexe, généralement plus compacte, visant la création de topologies structurées par pleiotropie.

### 2.3.2 *Codages directs*

De très nombreux systèmes ont été proposés pour coder directement un réseau de neurones. Des chaînes binaires ont été utilisées pour décrire une matrice de connexité (Miller et al., 1989) ou des listes de longueurs variables de neurones et de connexions (Cliff et al., 1993a) ; d'autres auteurs ont utilisé une séquence de paramètres, comme Pasemann et Dieckmann (1997), des suites d'entiers (Fullmer et Miikkulainen, 1991), un programme LISP (Koza et Rice, 1991) ou le graphe lui-même (Pan et Nie, 1996). Il est aisé, par exemple, de définir un génome comme une liste de longueur variable de triplets (poids, neurone-source, neurone-cible) à laquelle ont peut adjoindre une liste de biais, définissant ainsi les neurones. Dans la plupart des cas, la définition de l'opérateur de mutation est triviale. Pour des chaînes binaires, des bits aléatoirement choisis sont inversés. Pour des listes de paramètres, certains d'eux sont aléatoirement altérés. La définition d'un opérateur de croisement est néanmoins plus complexe à tel point qu'il n'est parfois pas utilisé du tout (Angeline et al., 1994; Pasemann et Dieckmann, 1997). Au delà de la complexité de mélanger efficacement deux graphes, le problème de la permutation (Radcliffe, 1993; Yao, 1999), dans lequel deux génotypes très différents représentent le même réseau, constitue un obstacle important à la création d'opérateurs de croisement fonctionnels. Par exemple, deux perceptrons multicouches avec des neurones cachés

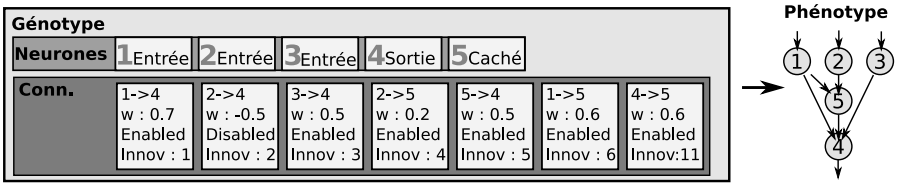



FIG. 2.7.: Le génotype de NEAT est constitué d'une liste de neurones et d'une liste de connexion. Un numéro d'innovation est affecté à chaque connexion. Les opérateurs génétiques peuvent insérer ou désactiver des gènes. D'après (Stanley et Miikkulainen, 2002a).

identique mais ordonnés différemment ont typiquement des génotypes différents mais représentent le même réseau. Un opérateur de croisement devrait alors tenter de trouver les meilleures correspondances entre les neurones, ce qui est difficile à définir si les réseaux n'ont pas le même nombre de neurones et qui s'avère computationnellement coûteux car une combinatoire importante doit être explorée.

NEAT (Neuro-Evolution of Augmented Toplogies ; Stanley et Miikkulainen (2002a)) apporte une solution originale au problème du croisement. Le point de départ de NEAT est que les gènes partageant le même historique dans le processus évolutionniste ont des chances d'être utilisés pour une fonction similaire. Afin de conserver l'historique génétique, un *numéro d'innovation* unique est attribué à chaque nouvelle connexion lors de son apparition via l'opérateur de mutation (figure 2.7). Ces marqueurs, conservés lors de la phase de reproduction, sont utilisés pour trouver les gènes homologues et ainsi aligner efficacement des génotypes de longueur différente. De plus, les numéros d'innovation permettent de définir une mesure de similarité computationnellement efficace entre les réseaux, contournant ainsi la NP-complétude de la distance d'édition (Bunke, 2000). Des réseaux sont d'autant plus similaires qu'ils partagent un nombre important de connexions avec le même numéro d'innovation. Cette mesure de similarité permet ensuite de calculer une distance pour créer des niches écologiques (Deb et Goldberg, 1989) afin de préserver la diversité de la population et, ainsi, de permettre à des structures nouvelles mais pas encore efficaces de survivre dans la population. La croissance incrémentale à partir d'une structure minimale constitue la troisième particularité de NEAT. Afin de réduire la dimension de l'espace de recherche, les auteurs proposent de commencer l'évolution à partir d'une unique topologie, typiquement un perceptron sans couche cachée, avec des poids synaptiques aléatoires. Cette idée contraste avec l'approche classique qui suggère de débuter le processus avec une population de topologies et paramètres aléatoires. Elle permet notamment d'éviter la présence de réseaux qui ne sont pas reliés aux entrées et aux sorties et de conserver des topologies simples. Ce processus d'initialisation limite l'utilisation de NEAT à des problèmes où il est possible de définir

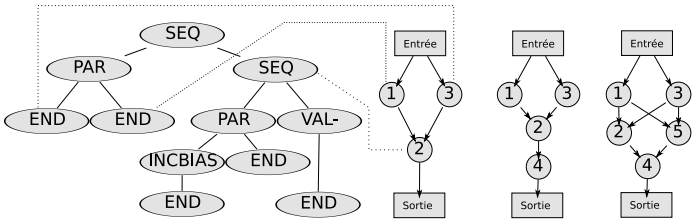



FIG. 2.8.: Génotype (gauche) et étapes de développement dans le cadre du codage cellulaire. L'exécution de l'instruction SEQ à la racine de l'arbre divise la cellule initiale 1 en cellules 1 et 2. L'instruction PAR divise ensuite la cellule 1 en cellules 1 et 3. Le processus d'évaluation de l'arbre se poursuit jusqu'à obtenir le réseau de neurones à l'extrême droite. D'après (Gruau, 1995).

un réseau de départ pertinent. NEAT a néanmoins été appliqué avec succès pour de nombreux problèmes tels le double pendule inversé (Stanley et Miikkulainen, 2002a), le contrôle de robots virtuels s'affrontant en duel (Stanley et Miikkulainen, 2004) ou de joueurs artificiels pour un jeu vidéo (Stanley et al., 2005).

### 2.3.3 *Codages indirects*

La simplicité des codages directs les rend attractifs mais beaucoup d'auteurs estiment qu'ils sont insuffisants pour faire évoluer des réseaux de grande taille. Pendant le développement biologique, le processus qui relie l'ADN au phénotype permet la réutilisation d'un même gène dans de nombreux contextes différents. Cette compacité permet de décrire des individus complexes avec comparativement très peu d'information. Pour reprendre une analogie proposée par Dawkins (1976), on décrit souvent le génome comme le plan d'un organisme mais il vaut mieux le voir comme une recette car il est irréversible, tout comme on ne peut retrouver facilement la recette d'un gâteau sortant du four en le goûtant. Les codages indirects s'inspirent grossièrement de ce principe en s'appuyant sur des représentations compactes qui sont, dans une deuxième phase, interprétées pour obtenir le phénotype en utilisant différents principes.

Ainsi, le système de génération de graphes proposé par Kitano (1990) s'appuie sur un ensemble de règles de réécriture d'une matrice de connexité. Poursuivant ces recherches, Gruau (Gruau, 1995) propose un « codage cellulaire » (*cellular encoding*) dans lequel le génome est constitué par des règles de réécriture des nœuds du graphe définissant le réseau de neurones (figure 2.8). Partant d'un réseau simple contenant des cellules reliées aux entrées et sorties du réseau, chaque cellule exécute un programme de développement. Une fois un symbole terminal atteint, la cellule est transformée en neurone. L'essentiel

des instructions dans ce langage de développement concerne les différentes manières de diviser les cellules et de créer les connexions lors des divisions. La représentation sous la forme d'un programme permet d'employer les opérateurs définis pour la programmation génétique (Koza, 1992), comme, par exemple, l'échange de sous-arbres. S'inspirant encore de la programmation génétique, Gruau ajouta par la suite des primitives pour créer automatiquement des fonctions (*automatically defined functions*, ADF). Cependant, comme l'ont souligné Luke et Spector (1996), échanger deux sous-arbres dans le codage cellulaire n'est pas équivalent à échanger deux sous-réseaux dans le phénotype car l'effet des instructions dépend de leur ordre d'exécution. De plus, le codage cellulaire n'inclut pas de méthode élégante pour faire évoluer les poids en plus de la topologie. Ces considérations amenèrent Luke (Luke et Spector, 1996) à proposer un codage aux mêmes inspirations mais centré sur les connexions au lieu des neurones. La même approche a été suivie par Hornby et Pollack (2002) qui ont exploré l'évolution corps-cerveau en combinant des L-systems (Lindenmayer, 1968) et le codage de Luke et Spector (1996).

### 2.3.4 *Codages géométriques*

La bonne connexion des entrées et sorties des réseaux de neurones est primordiale pour leur fonctionnement. Un réseau à la topologie parfaite mais connecté à la mauvaise sortie risque de ne pas fonctionner du tout ; or, quand le nombre d'entrées/sorties augmente, la probabilité d'effectuer une "mauvaise" connexion explose. Si l'on peut espérer que le processus évolutionniste effectue un tri efficace, il est illusoire que la génération aléatoire trouve les "bons" chemins quand le nombre d'entrées/sorties augmente significativement. Ce problème est particulièrement remarquable si le réseau prend une image en entrée, chaque pixel définissant une entrée, ou s'il est utilisé pour piloter des systèmes poly-articulés comme des robots à pattes. Afin de casser ce problème combinatoire, plusieurs auteurs ont proposé d'ajouter au processus de développement des codages indirects des informations géométriques ou topologiques : un neurone a plus de chance de se connecter aux neurones voisins qu'aux neurones éloignés. Le codage cellulaire a ainsi été étendu en situant les cellules dans un substrat géométrique en deux ou trois dimensions (Kodjabachian et Meyer, 1997), permettant ainsi aux instructions de développement de prendre en compte la position relative des neurones dans le substrat. Ce codage, baptisé SGOCE, a été appliqué avec succès à la génération d'un contrôleur pour un robot hexapode capable de marcher, d'éviter les obstacles et de suivre un gradient olfactif. Une idée proche a été exploitée dans (Cangelosi et al., 1994) où le génotype est constitué d'une liste de cellules contenant chacune des paramètres comme la longueur de l'axone ou l'angle de développement. Fukutani et Vaario (1997) se sont appuyés sur un autre schéma de développement qui prend explicitement en compte l'environnement, Dans son

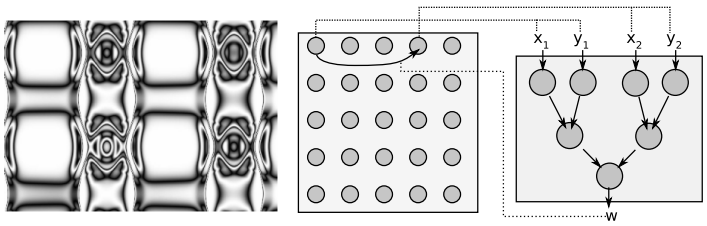



FIG. 2.9.: (a) Exemple d'image générée avec un CPNN avec deux entrées (les coordonnées (x,y) des pixels) et une sortie (le niveau de gris). (b) Principe d'utilisation d'un CPNN avec quatre entrées et une sortie pour générer les poids d'un réseau de neurones situé dans un substrat géométrique. D'après (D'Ambrosio et Stanley, 2007).

modèle, chaque cellule observe son environnement et, en fonction de celui-ci, choisit d'exécuter une règle de production.

HyperNEAT (D'Ambrosio et Stanley, 2007; Gauci et Stanley, 2007) suit une autre approche pour exploiter les régularités spatiales et la localité dans le but de faire évoluer de grands réseaux de neurones. NEAT est exploité pour faire évoluer des compositions de fonctions simples organisées sous forme de réseau, appelées Compositional Pattern Producing Networks (CPNN). De telles fonctions peuvent aisément créer des régularités dans leur espace de sortie (figure 2.9(a)) : une gaussienne ajoutera une symétrie, les fonctions périodiques des répétitions, etc. En créant de telles fonctions composées $f(x,y,z,w)$ possédant quatre entrées et une sortie, il est possible de déterminer le poids $w_{1,2}$ de la connexion reliant deux neurones dans le plan de coordonnées $(x_1,y_1)$, $(x_2,y_2)$ (figure 2.9(b)) : $w_{1,2} = f(x_1,y_1,x_2,y_2)$. Par ailleurs, il est possible avec cette méthode de générer des réseaux avec différents nombres de neurones mais conservant la même organisation globale. Bien que prometteuse, HyperNEAT n'a à ce jour été testé que sur quelques tâches simples sous la forme d'une expérience de vision simplifiée et du contrôle rudimentaire d'un robot mobile.

### 2.3.5 *Codages modulaires*

Les codages indirects présentés jusqu'à présent tentent d'exploiter les régularités des problèmes abordés en permettant, notamment, la répétition de structures et leur échange entre individus. Ces notions sont à rapprocher du concept de modularité tel que nous l'avons abordé dans la section 2.2.

Comme la modularité semble augmenter l'évolvabilité des systèmes complexes, il est particulièrement intéressant de tenter de l'incorporer dans les méthodes d'évolution de réseaux de neurones. Du fait de son génotype sous forme d'arbre de syntaxe, le codage cellulaire(Gruau, 1995) présente de nombreuses caractéristiques modulaires ; il est naturellement hiérarchique et pré-

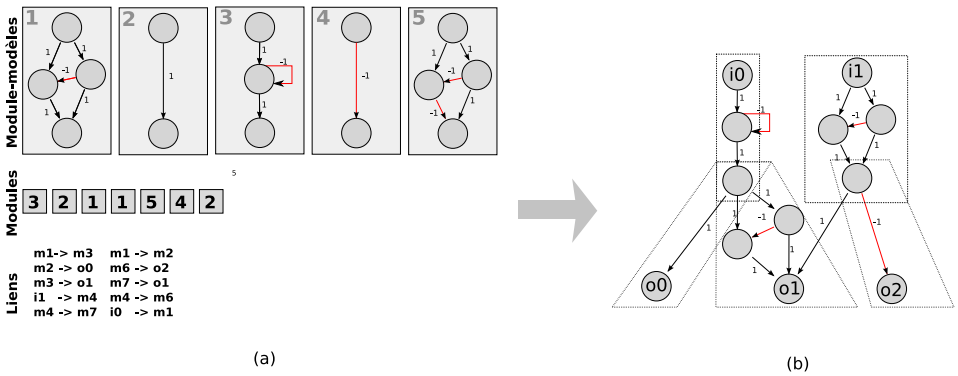



FIG. 2.10.: Principe du codage ModNet (Doncieux et Meyer, 2004a). Le génotype est constitué d'une liste de module-modèles, d'une liste de modules instanciés et d'une liste de liens entre les modules. Les modules-modèles peuvent être échangés par croisement et réutilisés plusieurs fois dans un même individu.

sente une structure forte. Cette dernière n'est cependant liée que de manière indirecte à la pression sélective. Afin d'incorporer la répétition dans le codage cellulaire, Gruau a proposé d'utiliser plusieurs arbres de syntaxe ordonnés qui sont interprétés de manière à ce que les nœuds terminaux d'un arbre soient des liens vers un des arbres suivants. Cette approche permet d'utiliser à plusieurs reprises un même sous-programme et donc un même sous-réseau. Le nombre d'arbres est imposé, contraignant beaucoup le processus.

Un mécanisme explicite de répétition combiné à un codage direct peut aussi être utilisé, comme le propose le codage ModNet (Doncieux et Meyer, 2004b). Un chromosome est constitué d'une liste de modules-modèles, une liste de modules et une liste de liens entre ces modules (figure 2.10). La liste de modules est constituée d'instances de modules-modèles qui peuvent ainsi être utilisés plusieurs fois dans le même réseau. Ceux-ci s'appuient sur un codage direct. Afin de simplifier la manipulation des modules, ces derniers ne peuvent avoir qu'une entrée et qu'une sortie. Le croisement est défini comme un échange de modules-modèles entre les chromosomes. Une des caractéristiques particulièrement intéressantes de ModNet est qu'il permet de donner au début de l'évolution des modules jugés potentiellement utiles créés à la main ou résultant d'une expérience précédente. ModNet été utilisé avec succès pour l'évolution de neuro-contrôleurs pour un pendule inversé, pour un dirigeable lenticulaire (Doncieux et Meyer, 2004b) et pour des animats à ailes battantes (Mouret et al., 2006). Néanmoins, l'absence de hiérarchie et la limitation du nombre d'entrées et de sorties de chaque module le limite à des problèmes de contrôle simples.

ModularNEAT (Reisinger et al., 2004) suit une approche similaire mais s'appuie sur NEAT pour l'évolution des modules. Ces derniers sont évolués symbiotiquement avec une population de « plans de montage » qui spécifie com-

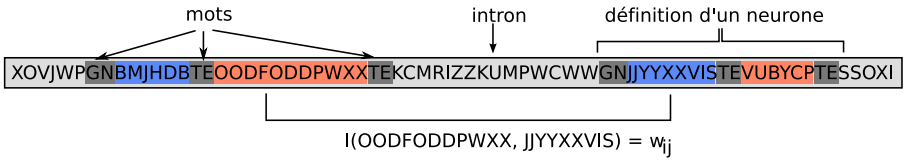



FIG. 2.11.: Principe du codage AGE (Mattiussi et Floreano, 2004). Le génotype est constitué d'une suite de caractères interprétés séquentiellement dont certains forment des mots. Dans le cas du codage d'un réseau de neurones, un neurone est défini par deux séquences correspondant respectivement à l'entrée et à la sortie du neurone. Une fonction d'interaction $I(x,y)$ est ensuite utilisée pour connaître l'influence de la sortie d'un neurone sur l'entrée d'un autre, c'est-à-dire le poids synaptique de la connexion les reliant.

ment les combiner, imposant une hiérarchie faite d'un seul niveau. Ce codage a été testé sur un jeu de plateau simplifié, montrant des performances significativement meilleures que celles obtenues avec NEAT.

L'utilisation de deux populations, une de composants et une d'individus les combinant, est aussi à l'origine de l'approche SANE (Moriarty et Miikkulainen, 1997) puis de MOBNET(Garcia-Pedrajas et al., 2005). Suivant le paradigme de la coévolution coopérative (Potter et De Jong, 2000), les composants, des neurones individuels dans SANE et des modules dans MOBNET, sont combinés en individus et différentes stratégies sont employées pour attribuer une fitness aux composants en fonction des performances des individus. Dans SANE, la fitness d'un neurone est définie comme la moyenne des fitness des réseaux auxquels il participe. Dans MOBNET, plusieurs critères du même ordre sont exploités à l'aide d'une approche multiobjectif.

### 2.3.6 *Codages implicites*

Dans les réseaux de gènes biologiques, l'interaction entre les gènes n'est pas explicitement encodée dans le génome mais elle est la conséquence de l'environnement physico-chimique dans lequel le génome est immergé. De tels réseaux régulateurs de gènes (gene regulatory networks, GRN), largement étudiés en génétique (par exemple, Teichmann et Babu (2004); Davidson et Erwin (2006)), semblent posséder d'importantes propriétés de modularité, de robustesse et de réutilisation. Plusieurs codages récents (Mattiussi et Floreano, 2004; Reisinger et Miikkulainen, 2007), baptisés codages implicites, tentent de s'en inspirer pour l'évolution de réseaux de neurones. Dans AGE (Analog Genetic Encoding) (Mattiussi et Floreano, 2004), le génome est constitué d'une suite de caractères (par exemple les caractères ASCII) interprétée séquentiellement et manipulée par des opérateurs génétiques simples (échange de sous-séquences,

insertion de caractères, etc.). Certaines suites de caractères forment des mots (*tokens*) qui, lors du décodage du génome, permettent d'interpréter la séquence suivante jusqu'au mot suivant (figure 2.11). Le mot représentant le début d'un neurone sera ainsi suivi de deux séquences délimitées par des mots, l'une associée à ses entrées ($s_1$) et l'autre à ses sorties ($s_2$). L'utilisation d'une fonction d'interaction $I(s_2^{(i)}, s_j^{(j)})$ permet de déterminer l'interaction, c'est-à-dire dans le cas des réseaux de neurones le poids synaptique, entre les neurones $i$ et $j$. AGE a été favorablement comparé à NEAT sur le problème du double pendule inversé dans (Dürr et al., 2006). Reisinger et Miikkulainen (2007) utilisent une autre abstraction des GRNs, basée sur des règles de production. Cette approche a pour l'instant été testée sur une variante du jeu Othello.

## 2.4 ROBOTIQUE ÉVOLUTIONNISTE

La robotique évolutionniste[2] (Cliff et al., 1993b; Meyer et al., 1998; Meyer, 1998; Nolfi et Floreano, 2001) correspond à l'application des algorithmes évolutionnistes à des robots pour mettre au point leur système de contrôle (leur « cerveau »), leur morphologie, ou les deux simultanément. Nous pouvons distinguer deux grandes catégories de recherches dans ce domaine, qui se distinguent plus par leur objectif que par les outils utilisés : l'étude des systèmes complexes et l'ingénierie automatique. Elles correspondent respectivement à l'utilisation de la robotique évolutionniste pour comprendre, d'une part, et pour créer, d'autre part. Ces deux approches ont mené à de nombreux résultats que nous récapitulons brièvement dans cette section. Nous présentons ensuite les deux principales voies proposées dans la littérature pour résoudre des problèmes plus complexes que ceux abordés jusqu'à présent : l'évolution incrémentale et la simulation des dynamiques inter-espèces. Enfin, pour des raisons pratiques, la plupart des travaux en robotique évolutionnistes s'appuient sur des robots simulés. Leur utilisation sur des robots réels soulèvent plusieurs problèmes que nous décrirons dans la dernière sous-section.

Suivant les approches, les auteurs s'autorisent à insérer dans le processus évolutionniste plus ou moins de connaissance, biaisant l'évolution vers certaines solutions. S'il parait normal de tenter de limiter la quantité de biais lors des études concernant l'émergence, cela peut aussi être intéressant dans le contexte de l'ingénierie automatique afin de laisser le maximum de latitude au processus évolutionniste pour mener aux solutions les plus originales. Dans cette optique, les réseaux de neurones ont été largement employés en raison de leur capacité à générer de très nombreux comportements différents et de leur relative plausibilité biologique. Compte tenu du propos de cette thèse, nous nous concentrons ici sur les méthodes les employant.

2 ou robotique évolutionnaire

### 2.4.1 *Étude des systèmes complexes*

La robotique évolutionniste peut s'inscrire dans le contexte de l'étude des systèmes complexes, c'est-à-dire des systèmes comprenant de nombreux composants interagissants dont les propriétés globales ne sont pas une conséquence directe de celles de leurs composants. Dans ce cadre, la robotique évolutionniste s'intéresse essentiellement à la « cognition minimale » (Harvey et al., 2005; Ruppin, 2002), c'est-à-dire aux processus adaptatifs les plus simples chez les animaux, et aux dynamiques évolutionnistes, telles la co-évolution entre deux espèces ou les interactions entre apprentissage et évolution. Les notions de progrès et d'efficacité apparaîssent alors secondaire au profit de l'analyse de l'émergence de comportements complexes à partir de l'interaction entre un robot et son environnement.

En tant qu'outil scientifique, la robotique évolutionniste ne peut générer que des preuves d'existence (Harvey et al., 2005). La réussite d'une expérience, c'est-à-dire l'observation dans un système obtenu par évolution artificielle de capacités adaptatives observées chez les êtres vivants ou de similarités entre les processus évolutionnistes artificiel et naturel, permet d'exhiber des conditions suffisantes à l'émergence de ces capacités et de ces dynamiques. La philosophie sous-jacente suppose de trouver les conditions suffisantes minimales, dans l'espoir de mieux comprendre celles qui sont nécessaires. Une expérience typique partira donc d'une théorie stipulant que l'existence du phénomène $X$ nécessite celle d'un phénomène $Y$. L'expérience peut réfuter cette théorie en montrant que $X$ peut exister sans $Y$ ou la préciser en permettant de comprendre dans quelles conditions $Y$ est nécessaire pour $X$. Les « véhicules » de Braitenberg (1984) forment un exemple stimulant de ce type de raisonnement : en montrant sur des robots mobiles que des comportements en apparence complexes – que l'on a tendance à interpréter comme de l'amour, de la peur, etc. – pouvaient être le résultat de l'interaction d'un contrôleur très simple avec un environnement complexe, Braitenberg a obligé à repenser l'interprétation du comportement des systèmes vivants. La robotique évolutionniste appliquée à la cognition tente d'automatiser ce processus en exploitant la puissance des algorithmes évolutionnistes pour trier les contraintes nécessaires de celles issues de conceptions parfois trop « humaines ». Afin d'y arriver, elle tente d'identifier et si possible d'éliminer tous les biais incorporés dans le processus évolutionniste.

Muni de ces concepts, le spectre des domaines abordés est très large (voir (Ruppin, 2002) et (Nolfi et Floreano, 2001) pour une revue détaillée), incluant notamment les liens entre évolution et apprentissage (Nolfi et Parisi, 1996), l'étude des *Central Pattern Generators* pour la locomotion (CPG, Ijspeert et Kodjabachian (1999); Chiel et al. (1999); Beer et al. (1999)), l'analyse des réseaux de neurones générés (Keinan et al., 2004) et la co-évolution entre plusieurs espèces. Nous présentons dans la suite de cette section quelques travaux ty-

piques pour chacune de ces problématique sauf la co-évolution, que nous aborderons en détail dans une section dédiée.

Bien que l'évolution envisagée dans un sens large puisse être vue comme une méthode d'apprentissage, on considère le plus souvent que ces deux processus adaptatifs opèrent à deux échelles de temps différentes, celle de l'espèce et celle de l'individu. Concrètement, dans une large majorité des travaux actuels de robotique évolutionniste, l'apprentissage modifie les poids synaptiques d'un réseau de neurones alors que l'évolution modifie la topologie du réseau et ou les mécanismes d'apprentissage. Néanmoins, comme le soulignent Yamauchi et Beer (1994), il est parfois difficile, voire biologiquement injustifié, de séparer les mécanismes d'apprentissage et ceux responsables du comportement car « beaucoup des mécanismes biochimiques mis en jeu sont identiques » (Yamauchi et Beer, 1994). Pour supporter cette hypothèse, Yamauchi et Beer ont fait évoluer puis ont analysé des réseaux récurrents d'intégrateurs à fuite. Ils ont pu exhiber des comportements avec l'apparence d'un apprentissage par renforcement alors qu'ils n'exploitaient que la dynamique des neurones pour changer leur comportement. Dans une perspective évolutionniste, l'apprentissage peut avoir différentes fonctions adaptatives (Nolfi et Floreano, 2001) :

- il permet aux individus de s'adapter aux changements de l'environnement qui apparaissent au cours de la vie d'un individu ;
- il peut aider et guider l'évolution, notamment via l'effet Baldwin (Baldwin, 1896; Ackley et Littman, 1992) : les individus qui apprennent vite sont d'abord sélectionnés, puis l'évolution pourrait favoriser les individus capables d'un comportement équivalent mais inscrit dans leur génotype, et donc non-appris, car l'apprentissage a un coût (il prend du temps par exemple) ; ce mécanisme correspond à une assimilation génétique indirecte des caractéristiques apprises ; des modèles computationels et robotiques ont été mis en place pour mieux comprendre ce phénomène (Hinton et Nowlan, 1987; Nolfi et al., 1994) ;
- il peut permettre la production de phénotypes complexes avec des génotypes simples en extrayant l'information nécessaire de l'environnement (Todd et Miller, 1991).

Notant qu'il est peu probable que le détail des poids synaptiques soit inscrit dans le génome, Floreano et Mondada (1996a) ont proposé de soumettre à évolution des règles d'apprentissage associées à chaque neurone. Ils ont testé cette approche dans une expérience où un robot roulant simulé devait naviguer en évitant les obstacles. Des contrôleurs fonctionnels ont été obtenus. Leur analyse révèle notamment que la connaissance dans ces réseaux de neurones n'est pas exprimée par une configuration finale stable mais par un point d'équilibre dynamique. Les synapses et leurs règles d'apprentissage sont alors responsables de l'apprentissage mais aussi de la régulation du comportement.

La recherche en robotique évolutionniste s'est aussi beaucoup penchée sur la locomotion animale et robotique en s'inspirant des *Central Pattern Generators*

(CPGs), des réseaux de neurones montrant un comportement oscillant même en dehors de toute stimulation et observés chez de nombreux animaux (Delcomyn, 1980; Hooper, 2001; Pearson, 2000; Marder, 2001). La locomotion animale est caractérisée par une activité rythmique et l'utilisation de nombreux degrés de liberté. N'importe quel insecte, par exemple, est expert dans le maniement de systèmes possédant de nombreux degrés de liberté dans un environnement aux lois complexes. Les CPGs (Delcomyn, 1980; Hooper, 2001; Pearson, 2000; Marder, 2001), souvent combinés avec un ensemble de réflexes semblent être à l'origine de ces performances mais la complexité des dynamiques qu'ils mettent en jeu les rendent difficiles à modéliser avec des méthodes classiques.

Des contrôleurs pour la marche de robots hexapodes ont ainsi été obtenus par évolution dans différents travaux en utilisant différents codages et méthodologies (Lewis et al., 1993; Gallagher et al., 1996; Jakobi, 1998; Kodjabachian et Meyer, 1997). Ils ont mené à l'observation de démarches tripodes similaires à celles utilisées par les insectes. Évolués en simulation, certains contrôleurs ont été transférés vers des robots réels (Kodjabachian et Meyer, 1997; Jakobi, 1998). Au-delà des insectes, les CPGs ont aussi été beaucoup étudiés chez la lamproie, un des vertébrés les plus simples. S'appuyant sur des résultats obtenus *in-vivo* par Ekeberg et al. (1995), Ijspeert et al. (1999) ont soumis à évolution les poids synaptiques et une partie de la topologie d'un CPG pour une lamproie simulée. Les CPGs obtenus couvrent un intervalle de fréquence plus large que le contrôleur créé manuellement, le rapprochant du CPG réel sans réduire la plausibilité biologique. De plus, les connexions évoluées se sont révélées, au moins dans certains cas, proches de celles mesurées *in-vivo*.

L'analyse des contrôleurs obtenus, le plus souvent des réseaux de neurones, par évolution est un point clef de la robotique évolutionniste dans le contexte de l'analyse des systèmes complexes, les réseaux obtenus étant le plus souvent peu structurés et difficile à interpréter. MSA (Multi-Perturbation Shapley Value Analysis, Keinan et al. (2004)) est un outil d'analyse récent, basé sur des résultats de théorie des jeux, qui permet de calculer la contribution d'un neurone ou d'une connexion à une fonction. Les auteurs se sont inspirés du principe des lésions utilisées par les biologistes pour comprendre les fonctions des groupes de neurones : un neurone ou un groupe de neurones est enlevé et l'on observe comment l'animal se comporte dans une situation déterminée. On en déduit l'importance de ce groupe de neurones pour réagir dans cette situation. Il est ensuite nécessaire de considérer toutes les combinaisons de lésions.

### 2.4.2 *Ingénierie automatique*

L'ingénierie automatique poursuit un but pragmatique : mettre au point des méthodes permettant de créer ou de contrôler des robots complexes de manière plus efficace que ce qu'il serait possible avec des méthodes d'ingénierie classiques. Les efforts dans cet axe se sont portés à la fois sur la génération de contrôleurs efficaces pour des problèmes non-linéaires, tels le double pendule

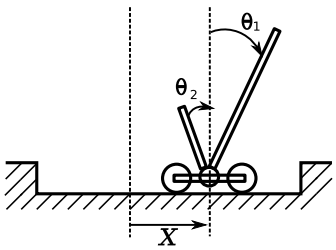



FIG. 2.12.: Stabilisation d'un double pendule inversé. Les pendules doivent être maintenus à la verticale en gardant le chariot le plus près du centre de la piste possible.

inversé, et sur le contrôle, éventuellement associé à la synthèse, de robots avec de nombreux degrés de liberté, tels les robots à pattes.

La stabilisation d'un pendule inversé (*cart-pole*) et ses extensions sont représentatives d'une large classe de systèmes instables non-linéaires. Elles sont devenues *de facto* le cadre de comparaison pour la plupart des codages de réseaux de neurones présentés à la section précédente (Wieland, 1991; Pasemann et Dieckmann, 1997; Stanley et Miikkulainen, 2002a; Geva et Sitte, 1993; Doncieux et Meyer, 2005; Gomez et al., 2006; Igel, 2003). Une barre – le pendule – est montée sur un chariot capable de se déplacer selon un axe. Le but est de maintenir le pendule vertical en bougeant uniquement le chariot dans certaines limites. Considéré comme trop simple dans certains travaux, ce problème a donné lieu à plusieurs extensions dont le double pendule inversé est certainement la plus populaire (figure 2.12). Dans ce cas, un deuxième pendule de taille différente est monté sur le même chariot. Les deux pendules doivent alors être maintenus à la verticale en ne bougeant que le chariot. Les contrôleurs à obtenir ont donc une ou deux entrées, selon la version du problème, et une sortie, le couple appliqué au moteur du chariot. En contrôle classique, un correcteur de proportionnel-dérivé (PD) résout efficacement ce problème. Deux variantes ont donc été traitée dans la littérature, avec et sans la dérivée de la position angulaire du pendule en entrée. Wieland (1991) et Pasemann et Dieckmann (1997) ont utilisé un codage direct pour contrôler un pendule avec et sans information de vitesse. Les résultats de NEAT ont été comparés à ceux obtenus avec le codage cellulaire et un codage direct (Stanley et Miikkulainen, 2002a), concluant à la supériorité de NEAT pour le double pendule inversé. Récemment, les performances du codage implicite AGE ont été confrontés avec succès à ceux de NEAT (Dürr et al., 2006). ModNet (Doncieux et Meyer, 2005) a aussi été testé avec succès sur le pendule inversé simple avec une fitness pénalisant fortement les oscillations mais le problème du double pendule n'a pas encore été abordé avec ce codage. Les « Echo State Networks »(ESN) ont également été utilisés avec succès pour une problématique similaire (Jiang et al., 2008). Les

ESN sont des réseaux de neurones récurrents dont toutes les connexions et les poids entre les neurones internes (cachés) sont fixés aléatoirement. Ce réseau dont la densité est fixée par l'utilisateur forme le *réservoir*. Seuls les poids des connexions reliant le réservoir aux sorties sont appris, par des procédures supervisées ou par évolution. Jiang et al. (2008) ont utilisé CMA-ES (Hansen et Ostermeier, 2001), une méthode évolutionniste particulièrement performante pour l'évolution de paramètres réels, pour déterminer ces poids afin de contrôler un double pendule inversé. Les auteurs obtiennent des performances comparable à celles de NEAT et de AGE.

Les robots modernes sont de plus en plus complexes et sont souvent de plus en plus difficiles à contrôler. Des approches évolutionnistes sont donc envisagées pour créer automatiquement des contrôleurs pour de tels robots, aux dynamiques parfois complexes et aux nombreux degrés de liberté. Parmi les méthodes précédemment décrites, NEAT a été utilisé pour générer des contrôleurs pour une fusée (Gomez et Miikkulainen, 2003) et ModNet a été employé pour la création de contrôleurs pour un dirigeable lenticulaire (Doncieux et Meyer, 2003a). Dans ce dernier cas, par exemple, l'évolution a su exploiter la forme du dirigeable pour monter plus vite que ce que l'on aurait obtenu avec un pilotage naïf. Des neuro-contrôleurs pour un drone à ailes battantes simulé ont aussi été obtenus avec ModNet (Mouret et al., 2006). Deux stratégies d'adaptation du battement d'ailes à des perturbations ont été trouvées, toutes deux efficaces. Afin d'économiser le temps nécessaire pour développer les programmes de marche du robot quadrupède AIBO, Hornby et al. (2000) ont utilisé un algorithme évolutionniste embarqué à bord du robot pour optimiser les paramètres du module de locomotion. Les démarches trouvées avec cette méthode sont plus rapides que celles créées manuellement. Plus généralement, le contrôle de robots à quatre, six ou huit pattes a fait l'objet de nombreuses recherches en robotique évolutionniste (Lewis et al., 1993; Gallagher et al., 1996; Jakobi, 1998; Kodjabachian et Meyer, 1997; Zykov et al., 2004). Kodjabachian et Meyer (1997) ont ainsi utilisé le codage SGOCE pour faire évoluer un réseau de neurones capable de piloter les différents degrés de liberté d'un robot hexapode de manière à ce qu'il soit capable de marcher mais aussi d'éviter les obstacles et de suivre un gradient, par exemple une odeur. Évolué en simulation, le contrôleur a été transféré avec succès sur un robot réel. Plus récemment, Zykov et al. (2004) ont utilisé l'évolution directement sur le robot réel dans le but de trouver des contrôleurs de marche dynamique pour un robot à pattes pneumatique, basé sur trois plateformes de Stewart. Des démarches efficaces ont été trouvées, les plus performantes étant les plus dynamiques, c'est-à-dire celles qui sont les plus difficiles à obtenir avec des méthodes de contrôle classiques. La marche dynamique bipède est probablement l'un des problème de locomotion les plus délicats. Paul et Bongard (2001) s'y sont intéressés en simulation en faisant évoluer un neuro-contrôleur de structure fixe. De manière étonnante, les performances des robots marcheurs se sont significativement améliorées lorsqu'ils ont laissé l'algorithme varier la distribution des masses.

Cette dernière expérience suggère qu'évoluer uniquement le contrôleur pourrait être trop restrictif et qu'il serait plus pertinent de faire co-évoluer la morphologie et le contrôle du robot, comme c'est le cas dans la nature. Dans un article séminal, Sims (1994) a ainsi fait évoluer des créatures à base de volumes géométriques simples dans un monde physique simulé en trois dimensions. Des couples d'individus sont en concurrence pour une ressource et les vainqueurs de chaque tournoi obtiennent une meilleure fitness ; ils peuvent donc ainsi survivre et se reproduire. La morphologie et l'architecture de contrôle, constituées de neurones, d'oscillateurs et de formes géométriques, sont toutes deux déterminées par le génotype, formé de graphes orientés. Elles évoluent donc de concert, soumises à une pression de sélection commune. Des créatures très diverses ont émergé du processus d'évolution, certaines possédant des morphologies et des stratégies d'action très similaires à celles observées dans le règne animal. En testant plusieurs stratégies de tournoi, Sims nota qu'il obtenait des résultats beaucoup plus intéressants en simulant plusieurs espèces en concurrence.

L'évolution morphologie-contrôleur a été plus récemment testée sur des robots faits d'un assemblage de barres et d'actionneurs (Lipson et Pollack, 2000). D'abord évalués en simulation, ces robots ont ensuite été construits à l'aide d'une machine de prototypage rapide. De nombreux types de robots fonctionnels ont émergés, montrant plusieurs caractéristiques surprenantes comme l'apparition de symétries, pourtant non spécifiées dans la fitness. Le codage direct utilisé dans ces expériences s'étant vite révélé limité, elles ont été poursuivies par des expériences similaires utilisant un codage indirect (Hornby et al., 2003) basé sur des L-systems (Lindenmayer, 1968). Les robots modulaires, capables de se reconfigurer sans intervention humaine, pourraient aussi constituer une approche prometteuse pour permettre l'évolution simultanée du contrôle et de la morphologie (Zykov et al., 2005).

### 2.4.3 *Le problème du* bootstrap *et l'évolution incrémentale*

Les fitness simples et peu directives, les plus séduisantes scientifiquement et techniquement, ne mènent pas toujours à des comportements intéressants en raison du problème du *bootstrap* : si tous les individus des premières générations obtiennent la fitness minimale, le processus évolutionniste ne peut pas démarrer et, par conséquent, aucun résultat significatif ne peut être trouvé. Par exemple, il a été jugé difficile d'obtenir un comportement de recherche de lumière dans une arène complexe si un réflexe d'évitement d'obstacle n'a été évolué précédemment (Urzelai et al., 1998). Plus généralement, il paraît illusoire de supposer que des individus au comportement intéressant émergeront nécessairement de l'étape de génération aléatoire pour tout problème complexe. Le problème du *bootstrap* se posant dès que l'on souhaite appliquer les concepts développés en robotique évolutionniste à des situations complexe, sa solution apparaît critique pour la progression de ce domaine de recherche.

Si l'on se penche sur la forme de l'espace de recherche, le problème du *bootstrap* correspond à un plateau, c'est-à-dire que tous les individus ont la même fitness et donc l'optimisation ne peut privilégier de direction particulière. Il est aussi possible de voir le problème sous la forme d'un minimum local particulier dans lequel aucune modification des individus de la population ne mène à de meilleures fitness. Les solutions apportées au problème du *bootstrap* s'appliquent donc souvent implicitement au problème des minima locaux, récurrent en optimisation.

Reconnaissant l'importance du problème du bootstrap, de nombreux chercheurs ont proposé des méthodes pour ajouter des étapes intermédiaires au processus, menant à des algorithmes d'*évolution incrémentale* (Harvey et al., 1994; Gomez et Miikkulainen, 1997; Kodjabachian et Meyer, 1997; Urzelai et Floreano, 1999). Cinq approches peuvent être distinguées : l'évolution par étapes (*staged evolution*), la complexification environnementale, le *fitness shaping*, la décomposition en comportements et la co-évolution.

Dans l'évolution par étapes, la tâche principale est décomposée en sous-tâches ordonnées, chacune associée à une fonction de fitness. Les individus sont d'abord sélectionnés en utilisant la première fitness. Une fois qu'un critère de convergence a été atteint pour une étape, par exemple quand une valeur minimum de la fitness a été atteinte ou quand la fitness ne change pas pendant quelques générations, l'expérimentateur change de fitness et passe à la tâche suivante. Cette technique a d'abord été employée avec succès par (Harvey et al., 1994) pour l'évolution de la localisation d'une cible en utilisant la vision. Trois étapes de complexité croissante ont été utilisées, de la localisation d'une cible grande et large au suivi d'une petite cible mobile. Beaucoup d'autres exemples peuvent être trouvés dans la littérature comme l'évolution de la marche d'un robot hexapode, puis de l'évitement d'obstacles, puis du suivi d'un gradient (Kodjabachian et Meyer, 1997) ; des problèmes de regression (Walker, 2004) pour comparer approche incrémentale et directe ; la marche hexapode où le problème du cycle de déplacement d'une patte est utilisé comme première étape (Parker, 2001) ; l'évolution d'un contrôleur de battement en boucle fermée pour un oiseau artificiel suivi de l'évolution d'un contrôleur de queue permettant d'effectuer des virages (Mouret et al., 2006).

La complexification environnementale permet de changer de manière plus continue la complexité de la tâche en modifiant des paramètres de l'expérience. Gomez et Miikkulainen (1997) ont ainsi travaillé sur une tâche où un prédateur, un robot roulant simulé, doit atteindre une proie. La simulation était paramétrée par la vitesse de la proie et par le délai avant de commencer la poursuite. Dix tâches ordonnées ont pu être définies par des valeurs spécifiques de ces paramètres et le processus change de tâche une fois qu'une valeur de fitness satisfaisante est obtenue pour la tâche intermédiaire. Ce processus incrémental a mené à des solutions plus efficaces qu'un processus monolithique.

La décomposition en comportements consiste à diviser le contrôleur du robot en sous-contrôleurs, chacun d'eux évolué séparément pour résoudre une

sous-tâche. Les sous-contrôleurs sont ensuite combinés ensemble à l'aide d'un deuxième processus évolutionniste. Cette approche a, par exemple, permis d'obtenir des contrôleurs composites pour un robot en Lego devant pousser une lumière dans une zone déterminée (Larsen et Hansen, 2005). Une variation de ce concept a été récemment utilisée pour faire évoluer un contrôleur capable de maintenir un hélicoptère autonome à une position fixe (De Nardi et al., 2006).

Le *fitness shaping* consiste à changer la fitness de manière à ce qu'elle récompense différemment les individus. Bien que cette approche ne soit généralement pas classée dans les méthodes d'évolution incrémentales, elle est souvent utilisée pour le *bootstrap* d'un processus évolutionniste. Dans son incarnation la plus typique, la fitness est définie comme une agrégation (une somme pondérée, par exemple) de différents critères d'évaluation afin de créer un gradient de fitness. Utilisant cette idée, Nolfi et Parisi (1995) ont réussi à obtenir un perceptron multicouche capable de piloter un robot pour localiser, reconnaître et attraper un objet. La fitness était augmentée si l'individu était près de l'objet, si l'objet était en face du robot, si le robot essayait d'attraper l'objet, si le robot avait l'objet dans la pince et si le robot relâchait l'objet au bon endroit.

### 2.4.4 *Co-évolution et fitness implicite*

L'utilisation d'une fonction de fitness explicitement liée à la performance des individus pour une tâche constitue une simplification importante du processus évolutionniste naturel. Si la nature ne rencontre pas le problème du *bootstrap*, c'est simplement qu'elle n'a pas d'objectif fixe ni de critère de performance explicite. Seule l'interaction entre les individus et leur environnement les pousse à trouver des stratégies originales pour supplanter les êtres vivants partageant la même niche écologique. En s'inspirant de cette idée, de nombreuses recherches ont été conduites dans lesquelles plusieurs espèces coévoluent dans un même environnement. De telles simulations ont été prolongées jusqu'à la mise en place d'écosystèmes artificiels complexes, en espérant laisser ainsi l'issue de l'évolution la plus ouverte possible. Qui peut prédire ce vers quoi une espèce évoluera pour éviter des prédateurs ? Comment deviner les stratégies que ceux-ci mettront en œuvre pour traquer leurs proies ? En basant ainsi la capacité de reproduction des individus sur les interactions inter-individus, inter-espèces et individu-environnement, le processus évolutionniste exploite une fitness *implicite*. La performance des individus n'est alors plus exprimée directement par l'expérimentateur comme dans les algorithmes évolutionnistes classiques. Elle émerge d'un système plus ouvert, idéalement capable de se complexifier à l'infini.

Les écosystèmes formés uniquement par une espèce de prédateurs et une autre de proies sont parmi les plus simples que l'on puisse envisager. Néanmoins, en mettant deux espèces en compétition, on peut espérer observer une « course aux armements » (Dawkins et Krebs, 1979) : si la proie devient plus effi-

cace, une pression de sélection importante poussera le prédateur à s'améliorer ; s'il réussit à combler son retard, c'est la proie qui devra trouver de nouvelles solutions ; le processus peut ainsi se répéter à l'infini. Afin de tester cette hypothèse, Nolfi et Floreano (1998, 2001) ont employé deux robots mobiles Khepera. L'un d'eux, le prédateur, était équipé d'un module de vision alors que l'autre, la proie, était aveugle et avait une vitesse de déplacement maximale égale à deux fois celle du prédateur. Chaque robot était contrôlé par un perceptron comprenant deux neurones cachés avec des connections récurrentes sur la couche de sortie. Les poids synaptiques étaient encodés par des chaînes binaires et soumis à évolution via un algorithme génétique classique. De nombreuses stratégies de chasse et de fuite ont été observées. Cependant, le même type de stratégie était redécouvert encore et encore, donnant lieu à des cycles. Comme le soulignent Rosin et Belew (1997), ces cycles peuvent parfois être évités en utilisant une archive des meilleurs individus rencontrés mais cette idée a peu de plausibilité biologique. Cependant, des résultats plus récents utilisant le codage de réseau de neurones NEAT (Stanley et Miikkulainen, 2002b) suggèrent que de bons résultats sont possibles si les génomes peuvent être progressivement complexifiés pendant l'évolution.

En s'éloignant de la métaphore biologique, il est possible de mettre au point des méthodes de co-évolution où une espèce constitue les élèves et une autre les professeurs. Chaque élève est sélectionné sur sa capacité à résoudre les exercices proposés par les différents professeurs, pendant que ceux-ci sont choisis en fonction de la pertinence des exercices proposés pour différentier les élèves. Une dynamique similaire à celle de la course aux armements peut alors s'enclencher : les professeurs commenceront par des exercices faciles car les élèves obtiendront tous la note minimale pour les plus difficiles, les rendant sans intérêt pour classer les étudiants ; ces derniers améliorant leurs compétences, certains nouveaux exercices, plus complexes, pourront s'avérer pertinents, stimulant à nouveau les élèves. Une telle dynamique pourrait résoudre efficacement le problème du *bootstrap* en adaptant la difficulté de la tâche aux performances des individus. Une première implémentation de ce principe à été proposée par Hillis (1991) pour la recherche d'algorithmes de tri. Une population d'algorithmes co-évoluait avec des listes d'entiers. Les premiers étaient récompensés en fonction de leur capacité à trier la liste alors que les listes l'étaient par rapport aux algorithmes qu'elles induisaient en erreur. En considérant chaque professeur, ou chaque test, comme un objectif, il est possible de classer les élèves en utilisant la relation de domination de Pareto. Dans ce cas, de Jong et Pollack (2004b) ont montré que l'on pouvait obtenir une évaluation « idéale » des individus. En robotique, ces tests pourraient être des situations différentes susceptibles d'être rencontrées par un robot ou des problèmes différents à résoudre. Malgré ses bonnes propriétés théoriques, l'utilisation en robotique évolutionniste de la co-évolution de Pareto se heurte aux mauvaises performances des algorithmes évolutionnistes multiobjectifs actuels au-delà de quelques objec-

tifs (De Jong et Bucci, 2008) et à la difficulté de définir un génotype pour des jeux de test en robotique.

Au lieu d'être en compétition, les différentes espèces peuvent aussi collaborer pour un même but. Dans la *co-évolution coopérative* (Potter et De Jong, 2000), des individus issus de chaque espèce sont ainsi combinés pour donner des solutions candidates au problème traité. Chaque espèce évolue séparément mais la fitness de chacun de leur membre dépend de leur contribution à la performance des individus composites. Bien que mettant en jeu une dynamique inter-espèce, ce type de co-évolution s'appuie sur une fonction de fitness explicite formulant le but commun aux espèces.

La co-évolution se concentre sur la pression que deux espèces s'exercent mutuellement quand elles partagent une niche écologique. Une autre idée pour obtenir des fitness implicites est de modéliser plus finement le processus de sélection naturelle en le situant dans un environnement. Un animal, par exemple, peut se reproduire plus vite que ses congénères, propageant ainsi ses gènes plus efficacement. Cependant, il dépensera beaucoup d'énergie ; épuisé, il pourrait être incapable de se reproduire l'année suivante ou de protéger sa progéniture, ce qui pourrait être néfaste à la dissémination de son patrimoine génétique. Plus généralement, chaque comportement animal est un compromis. Autrement dit, aucune caractéristique ne doit être directement maximisée : il est inutile de manger trop, au risque de se rendre malade ; il est absurde de courir vite si la dépense énergétique conduit l'organisme à la mort ; trop de repos ne présente pas plus d'intérêt. Ces compromis sont déterminés par l'interaction des individus avec leur environnement. Ils émergent d'un système complexe, créant de nombreux gradients de sélection simultanés. De plus, Le paysage de fitness est plus dynamique que celui induit par les les fonctions de fitness classiques car il dépend de l'ensemble des organismes habitant l'environnement.

Pour imiter la sélection naturelle, certains auteurs proposent de modéliser des agents situés dans un environnement simulé riche (Ray, 1991; Ackley et Littman, 1992; Yaeger, 1994; Channon et Damper, 1998; Taylor, 2001; Pichler et Canamero, 2007) dans lequel ils peuvent se déplacer, se nourrir, se rencontrer, se reproduire... Contrairement aux approches évolutionnistes classiques, la reproduction n'est pas liée à une notion de performance mais uniquement aux interactions entre les individus. Un agent se déplaçant vite rencontrera ainsi peut-être plus de partenaires ; un autre, mieux nourri, pariera peut-être sur sa durée de survie. Les individus sont alors libres d'utiliser différents moyens pour leur survie et il pourrait être possible d'atteindre des organismes artificiels au comportement plus riche que ce qui est possible avec les méthodes évolutionnistes traditionnelles. Néanmoins, la richesse de ce processus évolutionniste ouvert se traduit mécaniquement par une difficulté à diriger l'évolution vers la résolution de problème utiles. De telles approches sont donc intéressantes pour l'étude des systèmes complexe mais beaucoup moins pour l'ingénierie automatique.

De nombreux systèmes de ce type ont été étudiés dans le cadre de la « vie artificielle » (Langton, 1995). Dans Tierra (Ray, 1991), les individus sont des programmes informatiques en compétition pour la mémoire de l'ordinateur sur lequel ils sont exécutés. L'observation de l'évolution dans cet univers informatique a notamment montré l'émergence de parasites – des petites parties de code qui lancent la procédure de recopie d'un autre programme pour leur propre recopie. PolyWorld (Yaeger, 1994; Yaeger et Sporns, 2008) est un autre système écologique, s'inspirant plus directement de la nature afin de mieux en expliquer les dynamiques. Il est habité par une variété d'organismes qui s'appuient sur une vision simplifiée et sur des réseaux de neurones dont les poids sont déterminés par un apprentissage hebbien. Pour augmenter le réalisme des interactions, des expériences similaires peuvent être réalisées avec des robots. Ainsi, dans le projet Cyber Rodent (Doya et Uchibe, 2005; Elfwing et al., 2005), une population de robots doit maintenir un niveau d'énergie suffisant pour survivre dans un environnement où existent des batteries de rechange. Pour que son génome se propage, un robot doit trouver des batteries mais aussi des partenaires pour la reproduction et effectuer un échange de matériel génétique.

### 2.4.5 *Réalité et simulation*

Les approches systèmes complexes et ingénierie partagent les difficultés du passage de la simulation à la réalité (Nolfi et Floreano, 2001; Walker et al., 2003). Comme l'argumente (Brooks, 1992), il est possible que certains problèmes ne se posent qu'en simulation, et, inversement, que certains programmes fonctionnent très bien en simulation mais échouent dans le monde réel en raison de la complexité à simuler la dynamique de la réalité. Une simulation, aussi précise qu'elle soit, est nécessairement une version simplifiée du réel, excluant certaines interactions trop coûteuses à simuler ou simplement non comprises. Les choix de simplifications constitueront alors autant de biais susceptibles de détourner l'évolution de solutions pertinentes pour l'expérience. Même si de nombreuses hypothèses peuvent être testées en simulation, l'utilisation de robots réels renforce donc la pertinence de l'expérience.

Dans le cas de l'ingénierie automatique, le but est explicitement de mettre au point des robots plus performants. Le passage à la réalité est donc nécessaire. De plus, l'évolution est souvent utilisée dans l'espoir qu'elle puisse exploiter des couplages encore mal compris ou non triviaux. Ces couplages ont beaucoup plus de chances d'exister en réalité que dans une simulation. Par exemple, dans le cas de l'évolution de contrôleurs pour des robots inspirés des oiseaux (Doncieux et al., 2006), l'aérodynamique est très mal comprise et les simulateurs actuels sont très simplifiés. Les nombreux effets aérodynamiques non modélisés, qui pourraient expliquer les importantes différence entre les performances des oiseaux réels et simulés (de Margerie et al., 2007), ne pour-

ront être exploités par un robot volant que s'il prend efficacement en compte la réalité.

Une première approche pour combler le fossé entre simulation et réalité est de tenter de modéliser précisément les robots réels. Ainsi, Miglino et al. (1995) ont échantillonné les valeurs des capteurs de distance de robots Khepera pour les différents objets rencontrés, ce qui leur a permis d'intégrer dans leur simulateur des modèles de capteurs très réalistes et spécifiques au robot utilisé. Cette approche a été automatisée dans la robotique résiliente (Bongard et al., 2006) où le simulateur est évolué en parallèle d'un contrôleur embarqué dans un robot réel. La qualité de la simulation, le « modèle » du robot, s'accroît alors de générations en générations et peut même prendre en compte des événements comme la perte d'une patte du robot.

Bien que cela soit expérimentalement complexe, des expériences de robotique évolutionnistes ont été menées directement sur les robots réels. Ainsi, Floreano et Mondada (1996b) ont utilisé un robot Khepera dans une arène, relié à un ordinateur externe, pour faire évoluer un contrôleur permettant au robot d'éviter les obstacles et de se diriger vers une source lumineuse. L'expérience a duré dix jours d'évolution continue et de bons contrôleurs ont été obtenus. Plus récemment, Zykov et al. (2004) ont aussi directement utilisé un robot réel – un robot marcheur pneumatique à la dynamique complexe – pour obtenir des contrôleurs. Dans les deux cas, la grande majorité du temps a été passée à évaluer les contrôleurs, les robots ne bénéficiant malheureusement pas de la loi de Moore, contrairement aux simulations.

Une approche, plus radicale, a été avancée par Jakobi (1997). Selon la méthodologie proposée, seules les caractéristiques des interactions robot-environnement qui sont pertinentes pour l'émergence du comportement désiré sont simulées. Les autres sont changées aléatoirement pour s'assurer que les comportements s'appuient uniquement sur les caractéristiques de base. Cette méthodologie a été appliquée avec succès sur des robots marcheurs (Jakobi, 1998) et roulants (Jakobi, 1997). Cependant, elle nécessite l'identification des interactions pertinentes, ce qui peut être un biais significatif ou justement constituer la raison de l'utilisation d'une approche évolutionniste.

## 2.5 DISCUSSION

Le rapide panorama que nous venons de dresser des algorithmes évolutionnistes multiobjectifs, de l'évolution de réseaux de neurones, de la modularité et de la robotique évolutionniste révèle à quel point ces domaines pourtant très proches interagissent peu. Si l'intérêt de la modularité est perçu en robotique évolutionniste, peu de travaux s'appuient sur les résultats obtenus sur l'évolution de la modularité en biologie théorique. De même, la robotique pourrait largement bénéficier du concept de multiobjectif, par exemple pour modéliser des compromis performance/consommation énergétique, mais peu d'articles ont pour l'instant été publiés à ce sujet. Dans cette section, nous tentons de

mettre en valeur les difficultés des approches actuelles pour l'évolution de réseaux de neurones en robotique évolutionniste et d'exhiber les liens potentiels avec l'optimisation multiobjectif et la modularité.

### 2.5.1 *Comparaisons entre codages*

Si l'évolution de réseaux de neurones est beaucoup utilisée en robotique évolutionniste, le manque de comparaisons, de formalisations ainsi que la complexité des systèmes proposés rendent les progrès dans ce domaine difficile. Grönroos (1999) a comparé un codage direct, la grammaire de génération de graphe de Kitano (Kitano, 1990) et le codage basé sur un substrat géométrique de (Nolfi et Parisi, 1998). Deux problèmes ont été traités, une tâche de classification et le problème de l'encodeur. Leurs conclusions sont similaires à celles de Kitano concernant l'encodeur : la grammaire de génération de graphe mène à de meilleures performances. Cependant, le codage direct a obtenu les meilleures performances pour la tâche de classification. De plus, (Siddiqi, 1998) a montré qu'un codage direct correctement mis en place pouvait être aussi efficace que celui de Kitano sur l'encodeur. Le codage cellulaire a été comparé à un codage direct avec une topologie fixe pour le pendule inversé et ses variantes. Les deux codages ont donné des contrôleurs fonctionnels mais le codage cellulaire nécessite moins de connaissance pour être utilisé (Gruau et al., 1996). Dans (Stanley et Miikkulainen, 2002a), le nombre d'évaluations requis par NEAT a été comparé aux résultats numériques publiés dans (Gruau, 1995) pour les pendules inversés simple et double. Les auteurs ont conclu que NEAT nécessitait environ vingt cinq fois moins d'évaluations que le codage cellulaire pour des performances équivalentes. Récemment, Gomez et al. (2006) ont comparé plusieurs codages de réseaux de neurones pour les différentes variantes du pendule inversé. Les meilleurs résultats ont étés obtenus avec une topologie fixe, puis par NEAT. Le codage cellulaire se classe dernier. Enfin, AGE et les Echo State Networks semblent donner des résultats proches de ceux de NEAT pour le double pendule inversé (Dürr et al., 2006; Jiang et al., 2008).

Comme nous venons de le voir, la principale tâche utilisée pour comparer les performances des codages récents est la stabilisation du double pendule inversé. Introduite pour complexifier le pendule inversé en robotique évolutionniste, cette tâche n'est cependant pas exempte de défauts. Tout d'abord, il faut noter que le simple pendule inversé est un problème de contrôle classique, traité dans la plupart des cours de contrôle optimal, contrairement au double pendule. Le contrôle du pendule inversé nécessite un contrôleur proportionnel-dérivé (PD) bien réglé afin que le pendule atteigne la position cible rapidement et avec le moins d'oscillations possible. Obtenir des performances comparables à celles d'un tel contrôleur est beaucoup plus difficile que simplement maintenir le pendule vertical et la plupart des méthodes d'évolution de réseaux de neurones ne parviennent pas à égaler les contrôleurs PD (Mouret et Doncieux, 2008b; Doncieux, 2003). En outre, les caractéristiques du paysage de fitness du

double pendule inversé sont mal connues, rendant difficile l'appréciation de son intérêt. Le *bootstrap* du processus évolutionniste pour ce problème est notamment très difficile car très peu de configurations mènent à des fitness non nulles. Cette difficulté focalise les performances autant sur les biais dans la génération aléatoire et dans la pression de sélection que dans les capacités du codage. NEAT, par exemple, utilise des populations de 1000 individus et seuls quelques-uns obtiennent des fitness non nulles à l'issue de la génération aléatoire. NEAT étant fortement biaisé vers les réseaux *feed-forward*, en raison du réseau initial, un codage direct non biaisé aura probablement peu de chances de trouver au moins un réseau avec une fitness non nulle. Enfin, s'il est indéniable que le double pendule inversé est plus complexe que le simple, il reste à prouver qu'il est vraiment représentatif d'une classe de problèmes de contrôle intéressante. Notamment, il ne semble pas nécessiter de solutions modulaires, utilise peu d'entrées/sorties et requiert peu de capacités de « haut niveau » (comme, par exemple, une mémoire).

Si le double pendule inversé ne constitue pas un problème de comparaison idéal, peu de concurrents semblent faire consensus. Les tâches de traitement d'image, en raison de leur très grand nombre d'entrées pourraient être des candidates intéressantes pour tester le passage à l'échelle des approches envisagées. Néanmoins, une large majorité des tâches envisageables semblent ne nécessiter que des réseaux *feed-forward* comme en témoignent les nombreuses et fructueuses applications des perceptrons multicouches dans ce domaine (Bishop, 1995; Haykin, 1998). Les méthodes sans a priori sur la structure seraient alors d'un intérêt applicatif faible.

Les problèmes de panneaux routiers (*road-signs*, Rylatt et Czarnecki (2000); Yamauchi et Beer (1994); Ziemke et Thieme (2002)) forment une famille de tâches en robotique évolutionniste qui pourrait s'avérer plus intéressante. Un robot commence par voir une indication, généralement une lumière pouvant prendre deux couleurs. Il doit ensuite se diriger vers une zone de décision, par exemple un croisement, où il doit décider de sa direction en fonction de l'indication reçue au début de l'expérience. Les expériences de panneaux routiers nécessitent le stockage de la direction dans une mémoire, ce qui requiert des connexions récurrentes dans le réseau de neurones et élimine les comportements purement réactifs. Blynel et Floreano (2003) ont récemment exploré les capacités d'un réseau de neurones continus (des intégrateurs à fuite) complètement connecté pour résoudre un tel problème. Il est possible de construire des versions plus complexes comme la tâche Nid/Nourriture/Nid/Eau dans laquelle un robot doit successivement rejoindre les différentes zones dans le bon ordre. Capi et Doya (2005) ont récemment montré qu'un réseau de Elman (Elman, 1990) dont les poids sont soumis à évolution pouvait résoudre cette tâche, néanmoins aucune méthode où la topologie du réseau de neurones est soumise à évolution ne semble encore avoir été testée avec succès sur ce type de tâche.

Le peu de comparaisons peut aussi s'expliquer par l'absence de formalisme commun. Il est actuellement difficile d'implémenter dans un même contexte différentes méthodes et de les comparer théoriquement. Le formalisme proposé par Hussain (2003) est une tentative dans le but de réduire ce problème. Il est basé sur les grammaires à attributs (Knuth, 1968), une extension des grammaires hors-contexte qui ajoute la capacité de spécifier formellement la sémantique d'un langage. Chaque symbole est associé à un ensemble d'attributs et chaque règle de production à des règles d'évaluation de ces attributs. Les attributs sont divisés en deux ensembles distincts : les attributs *synthétisés*, qui sont le résultat des règles de mise à jour (calculés des feuilles à la racine de l'arbre de syntaxe) et les attributs *hérités* dont la valeur est issue du nœud parent (qui sont donc calculés de la racine vers les feuilles). De nombreux algorithmes ont été développés pour évaluer ces attributs (Paakki, 1995). Les opérateurs utilisés en programmation génétique sont utilisés pour faire évoluer des programmes spécifiés par une telle grammaire à attributs. Une grammaire à attributs peut être utilisée pour spécifier un langage de description de réseaux de neurones, un peu comme dans le codage cellulaire et Hussain a montré qu'une large classe de codage de réseau de neurones, notamment le codage cellulaire et ses variantes, pouvait être décrits par une grammaire à attributs. Cependant, des codages comme NEAT semblent difficiles à exprimer dans un tel formalisme en raison de leurs opérateurs particuliers et leur dépendance à l'algorithme évolutionniste, par exemple pour les niches.

### 2.5.2 *Modularité et réseaux de neurones*

Incorporer le concept de modularité dans les méthodes d'évolution de réseaux de neurones a depuis longtemps été reconnu comme nécessaire (Gruau, 1995). La multiplicité des approches envisagées par la biologie évolutionniste pour expliquer l'évolution des systèmes modulaires contraste étonnamment avec l'uniformité des approches employées pour ajouter la modularité aux réseaux de neurones. En effet, dans la grande majorité des cas (Gruau, 1995; Doncieux et Meyer, 2004a; Hornby, 2005; Reisinger et Miikkulainen, 2007), la modularité est inscrite dans le génotype. Après évolution, l'expérimentateur cherche à mettre en évidence des phénotypes modulaires et à savoir s'ils correspondent à des modules du génotype. La pression sélective menant éventuellement à la modularité n'est pas prise en compte. Implicitement, ces approches supposent que les génotypes modulaires rendus possibles par le codage modulaire seront sélectionnés car plus performants. Cette hypothèse correspond à la *sélection pour l'évolvabilité* (voir la section 2.2.4) contre laquelle de nombreux arguments biologiques militent (Wagner et al., 2005).

Si la modularité de la correspondance génotype/phénotype n'est pas remise en cause par la biologie, les modèles développés en biologie se concentrent exclusivement sur la pression de sélection, semblant négliger les caractéristiques du génotype. Plusieurs hypothèses font intervenir différentes pressions

directionnelles agissant sur des caractères différents, ce qui permet de relier la modularité phénotypique à celle du génotype par le biais de la sélection. Dans les méthodes d'évolution de réseaux de neurones étudiées, il est espéré que des fonctions distinctes émergeront dans le phénotype et que ces fonctions correspondront à des modules du génotype. L'existence de ce lien est le mystère même qui anime la biologie évolutionniste ; il paraît illusoire de le supposer comme acquis. Certains auteurs vont plus loin en montrant, grâce à des modèles théoriques, qu'il doit y avoir un « alignement » entre sélection, phénotype et génotype (Altenberg, 2005) :

> « Ma proposition principale est que les avantages évolutionnistes qui ont été attribués à la modularité ne dérivent pas de la modularité elle-même. Plutôt, ils requièrent qu'il y ait un "alignement" entre les espaces de variation phénotypiques et les gradients de sélection qui sont disponibles à un organisme. La modularité dans la correspondance génotype-phénotype peut rendre cet alignement plus facilement atteignable mais elle n'est pas suffisante ; la correspondance appropriée phénotype-fitness en conjonction avec celle génotype-phénotype est aussi nécessaire pour l'évolvabilité. »

Ces considérations militent fortement pour la prise en compte de la sélection dans l'évolution de systèmes modulaires, contrairement à ce qui a été proposé dans les méthodes actuelles d'évolution de réseaux de neurone.

### 2.5.3 *multiobjectif et robotique évolutionniste*

Probablement en raison de la relative jeunesse de l'évolution multiobjectif, les travaux importants en robotique évolutionniste se restreignent aux algorithmes mono-objectifs. En tant qu'instrument scientifique, les MOEA permettent d'analyser des compromis, par exemple la maniabilité d'un oiseau par rapport à ses performances en plané ou la dépense énergétique rapportée à la vitesse de déplacement d'un robot marcheur. Ce type d'analyse pourrait s'avérer particulièrement intéressant dans le contexte de la recherche sur les animats (Meyer, 1996, 1997b; Meyer et al., 2002). En tant qu'outil technique, les MOEA permettent d'optimiser plus efficacement et avec moins de paramètres les problèmes multiobjectifs, qui constituent la majorité des problèmes d'optimisation. Ils se comparent aussi favorablement à des algorithmes mono-objectifs sur des problèmes avec un unique objectif en permettant un meilleur maintien de la diversité Abbass et Deb (2003).

Ces qualités ont été exploitées dans quelques travaux en robotique évolutionniste. Teo et Abbass (2002) ont utilisé un MOEA pour faire évoluer un réseau de neurones pour un robot quadrupède en explorant le compromis entre maximiser les performances locomotrices et minimiser la complexité du réseau de neurones. Shan et al. (2000) ont pu mettre au point un CPG pour un robot humanoïde en optimisant le compromis entre équilibre, maintien de l'attitude et vitesse de marche. Dans des travaux antérieurs, nous avons opti-

misé le battement d'aile d'un robot à ailes battantes simulé en boucle fermée (Mouret et al., 2006) et en boucle ouverte (de Margerie et al., 2007). Le robot devait trouver des fonctions de battement capables de le maintenir à vitesse constante et à l'horizontale, avec et sans perturbation, tout en minimisant sa consommation énergétique. Ces différents travaux utilisent les MOEA comme technique d'optimisation en exprimant les caractéristiques des solutions souhaitées sous forme de plusieurs critères. Ils ne s'attardent pas sur les particularités de la robotique évolutionniste qui peut bénéficier de ce type d'approches dans d'autres situations, comme nous le verrons dans la suite de ce mémoire.

Les méthodes multiobjectifs étant issues des domaines de l'optimisation et de la décision, sa plausibilité biologique pourrait constituer un frein à son adoption dans des approches bio-mimétiques ou pour l'étude de l'évolution dans la nature. Le rapport entre l'évolution multiobjectif et la nature a étonnamment été peu étudié (voir Moshaiov (2006)) mais quelques idées préliminaires peuvent néanmoins être avancées à ce sujet. La relation de domination utilisée dans les MOEA aboutit ultimement à un classement entre les individus, la question est donc de savoir si la notion de compromis introduite par la relation de Pareto rend mieux compte de la capacité d'un être vivant à générer une descendance que celui de performance selon un critère unique, sa fitness.

Les compromis sont fondamentaux en écologie où ils représentent le « coût payé en terme de fitness lorsqu'un changement bénéfique d'un caractère est relié à un changement néfaste pour un autre » (Stearns, 1989). Le compromis entre la survie (les chances de survie de la mère et de sa progéniture) et la reproduction (la fréquence et la taille des portées) est notamment fondamental dans l'histoire d'une espèce. Plus généralement, la vie des êtres vivants est complexe et il existe de nombreuses manières pour un individu d'être meilleur qu'un autre. Ces différentes stratégies sont souvent incompatibles : par exemple, il n'est pas possible de bénéficier des avantages liés à une grande taille, comme pouvoir attraper des fruits en haut de arbres, avec ceux d'une petite taille, comme se cacher facilement. Un chat pourra courir plus vite, mieux récupérer après un effort, attirer plus facilement le sexe opposé, avoir un meilleur équilibre, un meilleur odorat, etc. Chacune de ces caractéristiques peut mener à des individus survivant mieux que leurs congénères. Avec le temps, certaines lignées peuvent se spécialiser en développant à l'extrême certains attributs, menant à de nouvelles espèces. Le guépard s'est spécialisé dans la course, la taupe dans l'exploration du sous-sol, etc. Des compromis peuvent aussi constituer des espèces capables de survivre car adaptées à une large variété de situations. Ces quelques considérations laissent supposer que la notion de domination peut avoir une plausibilité biologique et qu'elle pourrait même mieux représenter les différences entre les individus que la notion classique de fitness. Néanmoins, des études en biologie de l'évolution sont nécessaires pour mieux cerner les liens qui unissent domination et sélection naturelle.

# 3

# ÉVOLUTION INCRÉMENTALE ET OPTIMISATION MULTIOBJECTIF

RÉSUMÉ. Les algorithmes évolutionnistes ont été utilisés avec succès pour créer des contrôleurs pour de nombreux robots. Cependant, les fitness intuitives, par exemple le temps de survie du robot, ne permettent souvent pas de mener à des résultats intéressants en raison du problème du *bootstrap* : si tous les individus des premières générations obtiennent la fitness minimale, le processus évolutionniste ne peut pas démarrer et, par conséquent, aucun résultat significatif ne peut être trouvé. Pour surmonter cette difficulté, beaucoup d'auteurs ont proposé de définir des sous-tâches ordonnées, les amenant à définir un processus évolutionniste incrémental. Les méthodes publiées nécessitent une connaissance détaillée de la structure sous-jacente de la tâche analysée, qui n'est généralement pas accessible à l'expérimentateur. Dans ce chapitre, nous proposons d'assimiler chaque sous-tâche à un objectif puis d'utiliser un algorithme évolutionniste multiobjectif, créant ainsi un gradient de sélection pour chaque sous-tâche. Ce processus est capable de changer automatiquement entre les sous-tâches et ne requiert pas de les ordonner. La méthode proposée a été testée avec succès pour l'évolution des poids d'un perceptron multicouche pour une tâche de robotique mettant en jeu huit sous-tâches.

## 3.1 INTRODUCTION

COMME NOUS L'AVONS VU dans la section 2.4, le problème du *bootstrap* est récurrent en robotique évolutionniste et probablement présent dans beaucoup de tâches d'optimisation. Les méthodes proposées pour le résoudre s'appuient généralement sur une décomposition du problème en plusieurs étapes, définies par l'expérimentateur, afin de guider le processus. Ces approches ont mené à des schémas d'*évolution incrémentale* qui ont permis de créer par évolution des contrôleurs pour des tâches autrement inaccessibles (Chavas et al., 1998; Kodjabachian et Meyer, 1997; Mouret et al., 2006; Larsen et Hansen, 2005; De Nardi et al., 2006). Plus précisément, nous désignerons ici sous le terme de « tâche incrémentale » une tâche telle que :

– il n'existe pas de gradient de fitness dans la plus grande partie de l'espace de recherche et, notamment, la fitness de tous les individus tirés aléatoirement est minimum ;
– l'expérimentateur est capable de définir des sous-tâches plus simples dont la réussite est utile pour la réalisation de la tâche principale.

Malgré leur succès, les méthodes proposées (section 2.4) s'appuient sur de nombreuses suppositions qui requièrent une connaissance précise du problème traité et peuvent aisément mener le processus évolutionniste dans une impasse. L'évolution par étapes, le processus incrémental sans doute le plus largement utilisé, nécessite d'ordonner les différentes sous-tâches et de déterminer un critère de changement de sous-tâche. En outre, il n'est pas possible au processus évolutionniste de ne pas utiliser une sous-étape si celle-ci s'avère inutile, impossible ou contre-productive. Dans la décomposition comportementale, l'expérimentateur doit aussi choisir quand une sous-tâche est « résolue ». De plus, dans la décomposition comportementale, l'évolution par étapes et la complexification environementale, il n'existe plus de pression sélective explicite pour les sous-tâches une fois qu'elles sont considérées comme achevées. Dans la plupart des expériences, les contrôleurs issus des sous-tâches sont même fixés et ne peuvent donc plus être optimisés. Le *fitness shaping* est plus automatique mais impose aussi l'optimisation de toutes les sous-tâches – aucune ne peut être écartée – et, surtout, nécessite une mise au point laborieuse pour équilibrer l'importance de chaque sous-tâche lors de l'agrégation des différentes fitness. Ces nombreux biais et interventions de l'expérimentateur sont probablement une des raisons pour lesquelles ces approches n'ont été pour l'instant testées que sur des tâches ne mettant en jeu que deux à trois étapes. Des méthodes plus automatiques apparaissent nécessaires pour travailler sur des problèmes plus complexes afin de limiter le nombre d'hypothèses requises.

Ces difficultés peuvent être illustrées par l'évolution du vol chez les oiseaux. Imaginons que nous voulions créer par évolution des neuro-contrôleurs pour un robot à ailes battantes inspiré des oiseaux et capable d'effectuer des manœuvres complexes, par exemple de la voltige. Ce défi correspond notamment à celui du projet Robur (Doncieux et al., 2006; Mouret et al., 2006). Si l'on sélectionne les individus sur leur capacité à effectuer au mieux le programme de voltige (looping, tonneau, etc.), le défi est tel qu'aucun individu issu de la génération aléatoire n'obtiendra une fitness significative. Compte tenu de la difficulté à contrôler un simple vol horizontal (Mouret et al., 2006), il est fort possible qu'aucun individu ne parvienne à voler sur des trajectoires simples et à adapter son battement aux conditions rencontrées. Même si c'était le cas, il ne serait probablement pas sélectionné car une ligne droite est au moins autant éloignée d'un looping que la trajectoire effectuée par un robot battant des ailes n'importe comment. Néanmoins, on peut définir des étapes intermédiaires précédant la voltige : décoller, voler en ligne droite, effectuer des virages simples, .... Nous pouvons donc transformer notre problème en tâche

incrémentale et utiliser une méthode d'évolution incrémentale. Cependant, lorsque l'on défini les différentes sous-tâches, on s'aperçoit vite que leurs dépendances mutuelles sont inconnues. Il est donc très difficile de les ordonner et de savoir quand changer de sous-tâche. Ce problème trouve une illustration dans des travaux récents sur l'évolution de neuro-contrôleurs pour un robot à ailes battantes capable d'effectuer des virages (Mouret et al., 2006). Dans une première étape, deux contrôleurs de battement aux performances équivalentes mais à la stratégie très différente ont été mis au point avec un algorithme évolutionniste. Puis, suivant un schéma incrémental, ces deux contrôleurs pilotant les ailes ont été utilisés dans deux expériences différentes pour faire évoluer un second réseau de neurones indépendant actionnant la queue. Les performances en virage se sont révélées très dépendantes du contrôleur de battement employé alors que la fitness ne les distinguait pas lors de la première étape. Cette différence souligne les difficultés de l'évolution par étapes : deux solutions équivalentes à l'étape $n$ ne mènent pas forcément à des solutions aussi performantes à l'étape $n+1$.

Par ailleurs, il peut être souhaitable d'explorer plusieurs hypothèses pour une même étape. Si l'on cherche à s'inspirer de l'évolution des oiseaux réels, par exemple, de nombreuses hypothèses sont discutées pour expliquer l'apparition du vol (Padian et Chiappe, 1998). Les oiseaux ont pu commencer par sauter des arbres (hypothèse arboréale), par courir en s'aidant de leurs ailes sur le sol (hypothèse cursoriale) ou encore par s'aider de leurs ailes pour gravir des plans inclinés (Dial, 2003). De nombreuses autres variations de ces hypothèses ont été proposées. Comment commencer le processus évolutionniste pour l'évolution artificielle ? Quelle hypothèse favoriser ? Un processus d'évolution incrémentale devrait pouvoir essayer différentes hypothèses en parallèle avant de choisir la meilleure. Enfin, que se passe-t-il si deux sous-tâches doivent être développées de concert ? Par exemple, il paraît utile d'améliorer simultanément la forme des ailes et la cinématique de battement et non de définir une première étape pour la morphologie et une seconde pour le battement.

L'évolution de la plupart des comportements complexes animaux ou robotiques montrent des difficultés similaires à l'évolution du vol. Considérons par exemple les différentes sous-tâches que doit exécuter un rat pour survivre :

- marcher ;
- courir, ce qui implique peut-être de savoir marcher ;
- éviter les obstacles, ce qui nécessite de pouvoir se déplacer ;
- éviter les prédateurs, ce qui implique être capable de les percevoir et de mettre au point une stratégie de fuite ;
- manger, ce qui implique d'être capable d'explorer l'environnement et de localiser la nourriture ;
- se reposer, lorsque c'est nécessaire ;
- trouver un partenaire sexuel ;
- se reproduire (but final).

La réussite de toutes ces étapes a de bonnes chances de permettre à un rat de survivre. Il ne fait aucun doute qu'elles partagent des dépendances complexes ; presque toutes dépendent de la capacité à marcher, par exemple. Néanmoins, il paraît impossible de les ordonner *a priori* ni de savoir quel niveau minimum pour chaque sous-tâche est requis pour optimiser la suivante. De plus, il est *toujours* utile de savoir courir plus vite même si le processus de sélection se focalise sur une autre tâche. Une pression de sélection doit toujours être exercée sur les étapes précédentes même si des performances suffisantes pour passer à l'étape suivante ont été atteintes.

Si l'on regroupe les différents arguments développés jusqu'ici, un processus incrémental idéal devrait avoir les caractéristiques suivantes :

- le changement de sous-tâche doit être automatique, lorsque le processus le juge « approprié » ; il doit aussi être possible de faire marche arrière si le changement s'avère prématuré ;
- afin de contraindre le moins possible le processus, plusieurs hypothèses doivent pouvoir être explorées simultanément ;
- les dépendances entre sous-tâches doivent pouvoir être découvertes par le processus évolutionniste et non pré-supposées ;
- plusieurs sous-tâches doivent pouvoir être optimisées simultanément s'il s'avère que c'est la manière la plus efficace de procéder.

Aucune des méthodes proposées jusqu'à maintenant ne semble posséder ces caractéristiques. Dans ce chapitre, nous proposons une méthode basée sur l'évolution multiobjectif pour remédier à ces difficultés. L'approche proposée est testée sur une tâche de robotique évolutionniste simulée mettant en jeu huit sous-tâches et différentes hypothèse dans laquelle un robot doit successivement atteindre un ensemble de lumières dans une arène.

## 3.2 ÉVOLUTION INCRÉMENTALE ET OPTIMISATION MULTIOBJECTIF

L'évolution par étapes repose implicitement sur une modélisation de l'espace des fitness en forme de « L » (figure 3.1(a)). La première sous-tâche $T_s$ est optimisée en premier alors que la tâche principale $T_g$ ne l'est pas. La fitness de tous les individus sur la tâche principale peut alors prendre n'importe quelle valeur car elle ne correspond à aucune pression de sélection. En pratique, on peut s'attendre à ce que la fitness selon $T_s$ reste faible car les meilleurs individus ne sont pas favorisés. Une fois la valeur d'un critère de choix atteinte, la performance selon $T_s$ reste constante et celle selon $T_g$ s'améliore. En affinant l'analyse, il est possible que la performance selon $T_s$ change si le génome correspondant à cette sous-tâche n'est pas fixé ou que la performance selon $T_g$ influe sur celle selon $T_s$. Il est vraisemblable qu'elle décroisse peu car la nature incrémentale de $T_g$ suppose qu'une performance minimale selon $T_s$ est requise pour obtenir une fitness non-minimale selon $T_g$.

Comme nous l'avons précédemment évoqué, le choix de la génération où le changement de tâche est effectué est critique. S'il est accompli trop tôt, la

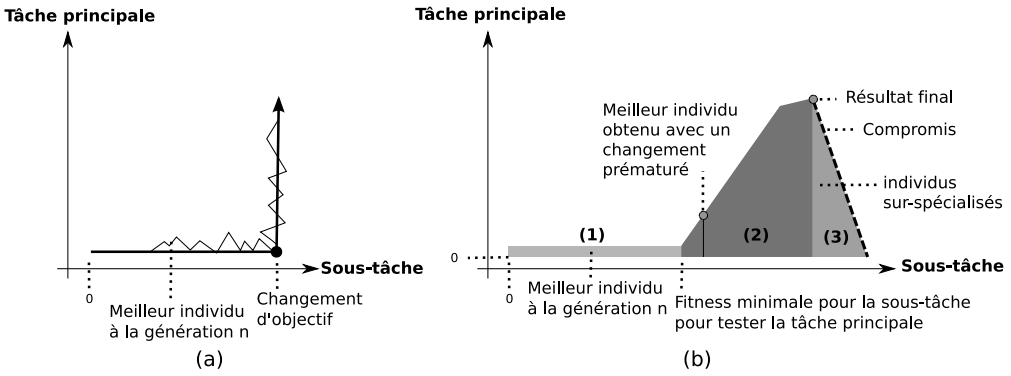



FIG. 3.1.: (a) Évolution par étapes lorsqu'une tâche principale dépend de la réussite minimale d'une sous-tâche. La sous-tâche est d'abord optimisée, la fitness selon la tâche principale pouvant prendre n'importe quelle valeur faible. À une génération donnée, l'expérimentateur décide de changer de fitness pour optimiser la tâche principale. (b) Espace objectif typique pour une tâche incrémentale. (1) Seule la première sous-tâche peut être optimisée ; (2) la fitness minimale pour essayer la tâche principale est atteinte, il est donc possible d'obtenir une fitness non-minimale pour la tâche principale ; (3) un compromis peut exister entre la sous-tâche et la tâche principale s'il est possible d'obtenir des individus sur-spécialisés.

sous-tâche ne sera pas complétement résolue et on peut s'attendre à ce que les performances pour la tâche principale, qui s'appuient dans ce cas sur une solution intermédiaire sous-optimale, soient moins bonnes que s'il avait été effectué plus tard. Inversement, si le changement est effectué trop tard, il est possible que les meilleures solutions selon $T_s$ soient sur-specialisées, rendant difficile leur intégration dans la solution finale. Ce dernier cas peut typiquement se produire si les tâches sont mal posées, c'est-à-dire si ce que l'on cherche n'est pas *exactement* ce qui est formulé dans la fitness. Il peut aussi se manifester si les ressources sont limitées, par exemple si le nombre de neurones possible est restreint pour l'ensemble des sous tâches. Dans ce cas, beaucoup de neurones peuvent être nécessaires pour obtenir la meilleure fitness selon $T_s$, ne laissant aucun neurone disponible pour résoudre la tâche principale. Il est donc possible, suivant les problèmes traités, qu'il existe un *compromis* entre les tâches.

La figure 3.1(b) résume graphiquement notre hypothèse sur la forme de l'espace des fitness dans les tâches incrémentales. Pour une sous-tâche $T_s$ et une tâche principale $T_g$, nous supposons qu'il se divise en trois zones :

1. une zone où aucun individu ne peut avoir de fitness non-minimale selon $T_g$ et où l'optimisation a donc uniquement lieu selon $T_s$ ;

2. une zone où les individus ont dépassé la fitness minimale selon $T_s$ correspondant au minimum requis pour obtenir une fitness non-minimale selon $T_g$ ;

3. une zone où les individus sont sur-spécialisés, c'est-à-dire qu'ils ne peuvent pas obtenir à la fois la fitness maximum selon $T_g$ et selon $T_s$.

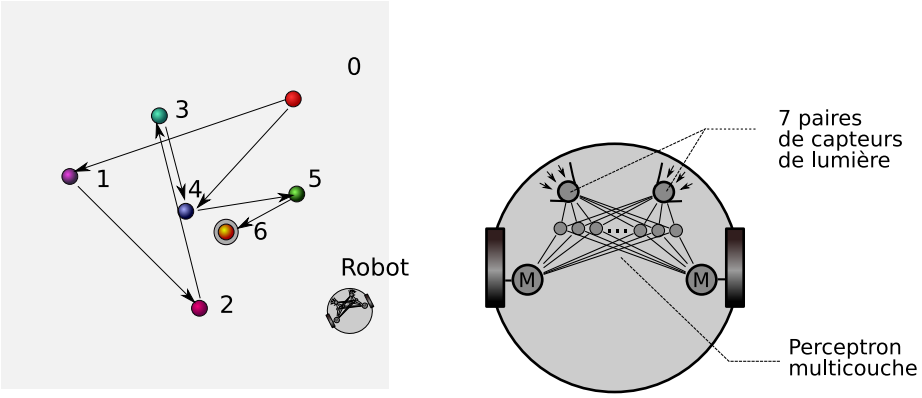
Ces trois zones générales ne sont évidemment pas nécessairement présentes dans tous les problèmes incrémentaux. De plus, cette hypothèse se généralise aisément à plus de d'une sous-tâche et à des liens plus complexes entre sous-tâches. Une méthode d'évolution incrémentale idéale devrait donc optimiser selon les sous-tâches possibles (zone 1) puis, automatiquement, explorer simultanément les différentes tâches accessibles (zone 2) et, enfin, gérer le compromis lié à la sur-spécialisation (zone 3).

Si l'on considère chaque tâche – sous-tâches et tâche principale – comme des objectifs, c'est exactement ce que fait un algorithme évolutionniste multiobjectif (MOEA, section 2.1.2). Un tel algorithme commencera par essayer toutes les tâches simultanément. Si la tâche principale a une structure incrémentale, seul un petit sous-ensemble des tâches devrait être réalisable au début du processus. Comme la fitness selon les autres sous-tâches sera minimum, la sélection se fera uniquement selon ces premières sous-tâches, les plus simples ; l'évolution devrait alors pouvoir obtenir progressivement des individus de plus en plus performants pour ces sous-tâches. Si un individu atteint des performances suffisantes pour tester une autre sous-tâche, il sera immédiatement non-dominé et donc sélectionné pour la reproduction. L'évolution tentera alors d'explorer cette tâche en parallèle des autres. En raison des méthodes de répartition des solutions sur le front de Pareto, si la population contient déjà beaucoup d'individus performants sur les anciennes tâches, le processus peut se concentrer sur la nouvelle. Le changement d'objectif s'est alors effectué automatiquement. En outre, un MOEA sera capable d'explorer plusieurs hypothèses, car l'optimisation selon un des objectifs n'empêche pas celle des autres.

L'évolution incrémentale peut donc être vue comme un cas particulier de l'optimisation multiobjectif où les différents objectifs sont liés et où l'on ne s'intéresse au final qu'aux individus performants selon un sous-ensemble d'objectifs. L'ensemble des algorithmes vu dans la section 2.1.1 est donc susceptible d'être directement utilisé pour résoudre les tâches incrémentales.

L'agrégation entre objectifs par une somme ou un produit étant une méthode pour réaliser une optimisaton multiobjectif, le *fitness shaping* comme l'utilisent Nolfi et Parisi (1995) peut être vu comme une approche similaire à celle que nous venons de présenter. Néanmoins, au moins trois aspects différencient notre approche de la leur :

– suivant la forme du front de Pareto, certaines solutions ne peuvent pas être trouvées avec une méthode d'agrégation simple ; ces solutions peuvent pourtant être utiles au processus évolutionniste ;




(a) (b)

FIG. 3.2.: (a) Structure de la tâche de phototropie. Sept lampes colorées sont montées sur des interrupteurs qui allument d'autres lampes dans l'arène. Le circuit électrique, inconnu de l'algorithme, et représenté par des flèches. La tâche finale est d'atteindre la lampe 6. (b) le robot simulé est équipé de sept paires de capteurs de lumière, chacun d'eux étant sensible à une couleur particulière. Le contrôleur est un perceptron multicouche avec 15 entrées (une par capteur et un biais), 15 neurones cachés et 2 sorties (la vitesse des moteurs).

– le choix des poids entre chaque objectif est difficile, surtout lorsque beaucoup d'étapes sont en jeu, et fixe implicitement le chemin que doit suivre le processus ;
– les meilleurs individus pour chaque sous-tâche ne sont pas conservés, rendant impossible l'exploration simultanée de plusieurs chemins entre les tâches ; par exemple, une approche basée sur l'agrégation peut être piégée dans une zone de sur-spécialisation.

Sous une forme plus générale, on peut voir l'approche basée sur l'agrégation comme une version gloutonne de l'approche multiobjectif, où toutes les sous-tâches possibles sont optimisées au mieux mais où il est impossible de faire marche arrière ou d'explorer plusieurs hypothèses. Par ailleurs, une fois reconnu que l'évolution incrémentale correspond à une optimisation multiobjectif, il parait naturel d'utiliser des algorithmes prévus pour cela, aux bonnes propriétés théoriques et aux bonnes performances, alors que l'agrégation sous forme de somme ou de produit est unanimement reconnue comme une mauvaise méthode pour optimiser plusieurs objectifs (Deb, 2001).

## 3.3 ROBOT PHOTOTROPE

### 3.3.1 *Expérience*

Afin de tester cette approche et de valider expérimentalement l'hypothèse que nous faisons sur la structure de l'espace des fitness, nous avons mis au point une variante de la tâche classique de recherche d'une lumière par un robot dans une arène, par exemple utilisée par (Floreano et Urzelai, 2001) pour étudier les liens entre évolution et apprentissage. Le robot étant attiré par la lumière, nous le désignerons sous le terme de « robot phototrope ». La tâche choisie met en jeu sept sous-tâches avec des dépendances complexes.

Dans la variante étudiée ici, un robot simulé est placé dans une arène avec sept lampes colorées différentes (figure 3.2(a)). Chacune d'elle, initialement éteinte, est montée sur un interrupteur qui, lorsque le robot appuie dessus, allume une ou plusieurs lampes dans l'arène. Une fois allumée, une lampe le reste jusqu'à la fin de l'expérience. Le but principal est d'allumer une lampe particulière. L'allumage d'une lampe quelconque définit une sous-tâche. Les connections entre les interrupteurs modélisent donc les dépendances entre les différentes tâches ; elles sont inconnues de l'algorithme évolutionniste. La structure retenue dans notre expérience est représentée sur la figure 3.2(a). La première lampe active deux autres lampes, créant deux chemins pour accomplir la tâche finale. Une partie de la population peut choisir le chemin $\{0,1,2,3,6\}$ alors qu'une autre peut choisir le chemin, plus court et comportant moins d'étapes, $\{0,4,5,6\}$.

Le robot simulé (figure 3.2(b)) est équipé de sept paires de capteurs de lumière, chacune d'elle étant sensible à une couleur différente. Sur un vrai robot, ces capteurs de lumière pourraient être remplacés par une caméra. Chaque capteur a un champ de vision de 90 degrés et une sortie binaire, 1 si la lampe est dans le champ de vision, 0 sinon. Le robot est contrôlé par un simple perceptron multicouche avec 14 entrées, 2 sorties et 14 neurones cachés. Ces derniers utilisent la fonction d'activation suivante, définie pour permettre l'inhibition d'un neurone à l'aide d'entrées négatives :

$$f(x) = \begin{cases} \tanh(x) \text{ si } x > 0 \\ 0 \text{ sinon} \end{cases}$$

Seul les poids synaptiques sont soumis à évolution via l'utilisation d'une mutation gaussienne. La recombinaison (*cross-over*) n'est pas utilisée.

La seule connaissance utilisée pour le *bootstrap* du processus est donc que le fait qu'allumer certaines lampes devrait permettre d'atteindre la lampe objectif. Malgré la simplicité des tâches, le robot doit accomplir au moins quatre sous-tâches (le chemin le plus court), en explorant deux hypothèses différentes. Cela fait de cette tâche de « phototropie incrémentale » l'un des problème les plus complexes abordés jusqu'à présent avec l'évolution incrémentale. Au delà de sa complexité, cette tâche est intéressante pour au moins quatre raisons :

– elle est incrémentale parce que la réussite de la tâche principale passe nécessairement par la réussite d'une partie des sous-tâches et parce qu'aucun individu ne peut découvrir « par hasard » la bonne séquence au début d'une expérience ;
– elle peut néanmoins être résolue avec un contrôleur simple *feed-forward* inspiré des véhicules de Braitenberg ; sa difficulté réside donc *uniquement* dans la pression de sélection ;
– elle permet de modéliser plusieurs hypothèses, chacune d'elle correspondant à une séquence différente pour allumer la dernière lampe ;
– sa difficulté peut être changée en ajoutant / enlevant des lampes ou en modifiant les dépendances entre les lampes.

### 3.3.2 *Fitness et algorithme évolutionniste*

Un algorithme évolutionniste standard avec une fitness conçue pour minimiser le nombre de pas de temps nécessaires pour allumer la dernière lampe ne trouve pas de contrôleur fonctionnel, tous les individus obtenant une fitness égale au nombre de pas de temps de l'expérience. Nous sommes donc bien en présence d'une tâche incrémentale.

Pour utiliser l'approche proposée, sept objectifs (à maximiser) sont définis. Ils correspondent à l'opposé du nombre de pas de temps depuis le début de l'expérience requis pour actionner chaque interrupteur. Pour éviter le surapprentissage, chaque objectif est le score minimal pour trois expériences dans lesquelles le robot démarre à des positions différentes. Soient $\varphi(n,i)$ le nombre de pas de temps passés avant d'allumer la lampe $i$, pour l'expérience $n$ et $T$ le nombre de pas de temps total. Chaque objectif $F_i$ est alors calculé de la façon suivante :

$$F_i = \min_{n=1,2,3} \frac{-\varphi(n,i)}{T}, i \in 0,1,\cdots,6$$

L'expérience a été lancée 30 fois avec l'algorithme NSGA2 (Deb et al., 2000a), 200 individus et pendant 1000 générations. Des résultats similaires ont été obtenus avec $\varepsilon$-MOEA (Mouret et Doncieux, 2008a).

### 3.3.3 *Résultats*

Toutes les expériences lancées ont permis d'obtenir des contrôleurs guidant le robot sur le chemin le plus court et atteignant la dernière lampe. La figure 3.3(a) présente un exemple de front de Pareto obtenu à la fin d'une telle expérience. Les différentes stratégies possibles sont visibles : certains individus, les plus performants pour le dernier critère, suivent la séquence $\{0,4,5,6\}$, alors que d'autres suivent $\{0,1,2,3,4,5,6\}$. Une part des individus non-dominés n'atteignent pas la lampe 6 car ils représentent les meilleures solutions obtenues pour atteindre une lampe particulière ou pour effectuer une séquence

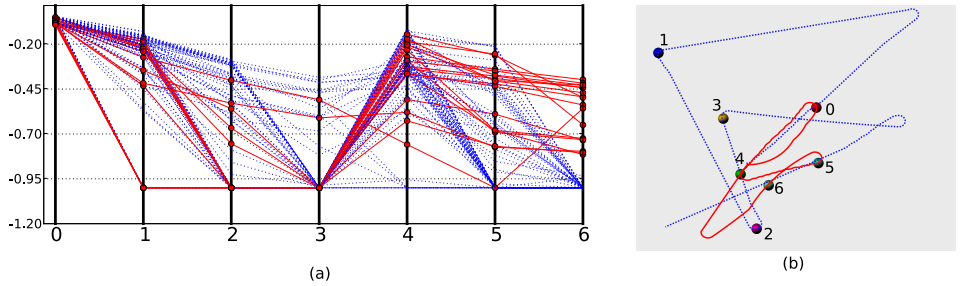



FIG. 3.3.: (a) Diagramme parallèle des individus du front de Pareto après 1000 générations. Chaque axe vertical correspond à un objectif et un individu est représenté par un chemin reliant sa valeur pour chaque objectif sur chaque axe. Les lignes pleines correspondent aux individus qui résolvent la tâche, c'est-à-dire qui atteignent la lampe 6 avant la fin de l'expérience. (b) Deux exemples de trajectoires obtenues atteignant la lampe 6. Chacune correspond à un compromis différent.

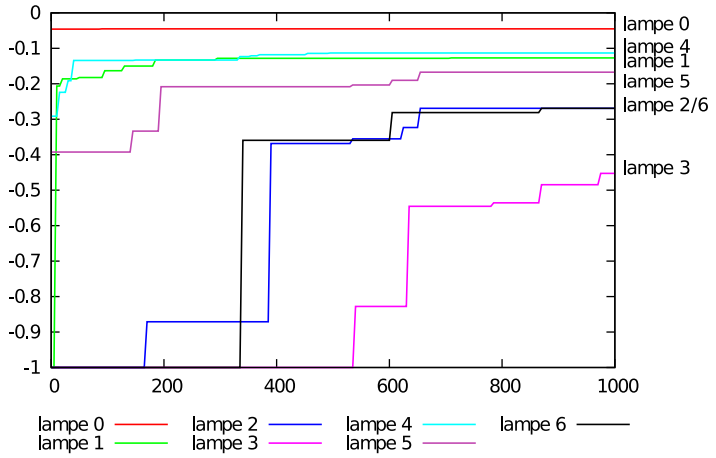



FIG. 3.4.: Meilleure fitness obtenue pour chaque objectif en fonction de la génération, pour une expérience typique. Les lampes 0, 1 et 4 sont d'abord atteintes. Elles sont suivies par la lampe 2 (génération 180) et la lampe 6 (génération 370).

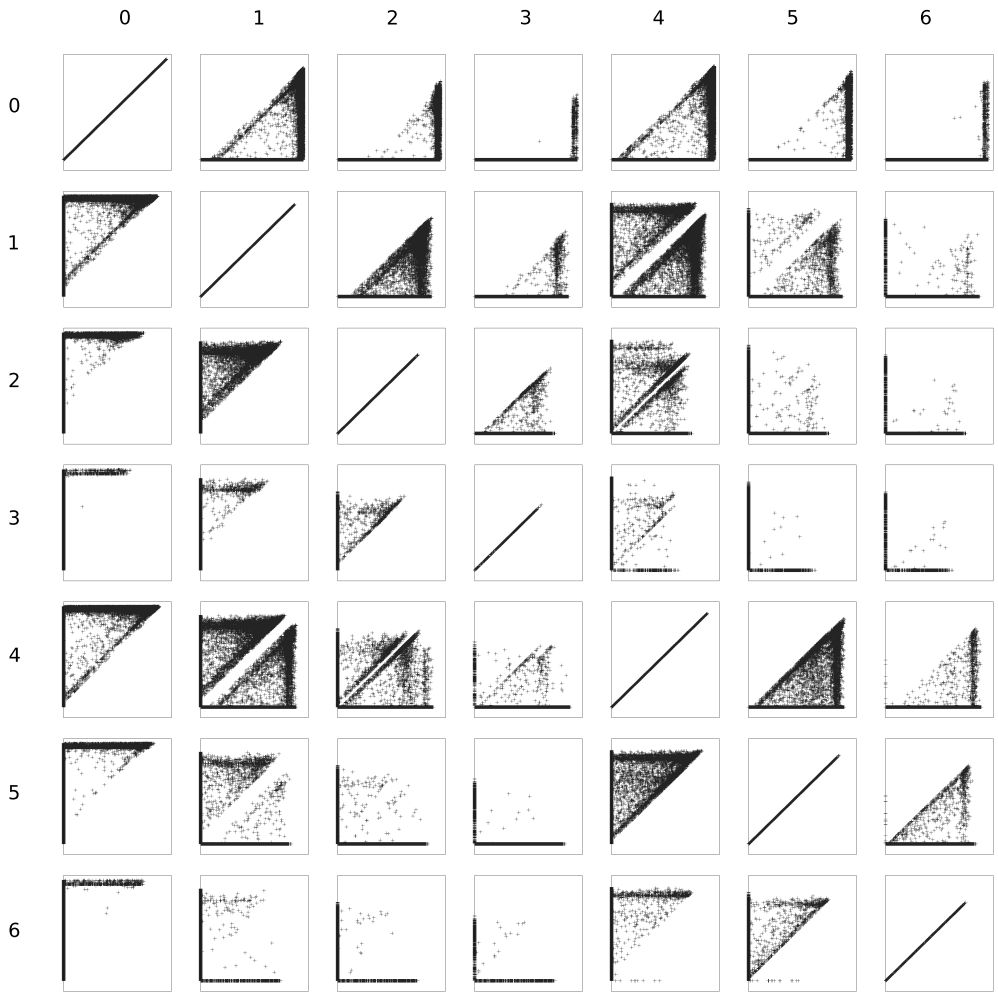
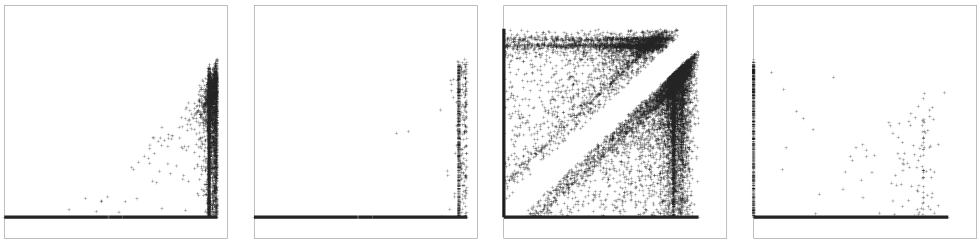



FIG. 3.5.: Fitness obtenues par tous les individus pendant une expérience. Le graphe de la ligne $i$ et de la colonne $j$ correspond à l'objectif $j$ en fonction de l'objectif $i$. Ces graphes révèlent les dépendances entre objectifs. La figure 3.6 présente un agrandissement de certains d'eux.



(a) 2 en fonction de 0 (b) 6 en fonction de 0 (c) 4 en fonction de 1 (d) 6 en fonction de 1

FIG. 3.6.: Agrandissement de certains graphes de la figure 3.5. (a) Objectif 2 en fonction de l'objectif 0. On distingue une zone de type "0" et une zone de type "1". (b) Objectif 6 en fonction de l'objectif 0. Tous les individus obtenant un bon score pour l'objectif 6 ont du résoudre efficacement l'objectif 1, on observe donc une forme de "L". (c) Objectif 4 en fonction de 1. Puisque la lampe 0 allume la lampe 1 et 4, il existe deux ordres possibles pour actionner l'interrupteur 4, avant et après le 1. Le motif observé sur cette figure reflète ces deux possibilités. (d) Objectif 6 en fonction de l'objectif 1. Il est possible d'atteindre la lampe 6 sans passer la lampe 1, on observe donc peu de dépendances entre les deux objectifs.

donnée n'incluant pas celle-ci. Deux exemples de trajectoires obtenues sont présentés sur la figure 3.3(b).

La figure 3.4 illustre la progression par étapes du processus évolutionniste. Elle correspond à la meilleure valeur observée parmi les individus du front de Pareto pour chaque objectif et chaque génération. Le chemin suivi par l'évolution est en accord avec l'intuition : des fitness non-minimales pour chaque objectif sont progressivement atteintes, l'ordre étant déterminé par le circuit reliant les lampes. Ainsi, dès les premières générations, des contrôleurs atteignent les lampe 0,1, 4 et 5 alors qu'aucun ne touche la lampe 6. 180 générations sont nécessaires pour atteindre la lampe 2 et 370 pour la lampe 6, l'objectif final. En outre, les performances pour chaque sous-tâches continuent de progresser de générations en générations. La lampe 3 est étonnamment atteinte la dernière, ce qui s'explique par le fait qu'il y a autant d'étapes pour rejoindre la lampes 3 que la lampe 6 mais que le chemin spatial est plus long (figure 3.2 (a)).

Il est possible d'analyser les dépendances entre objectifs et la forme de l'espace des fitness en plaçant sur un graphe multi-dimensionel chaque individu testé par l'évolution. Puisque nous avons utilisé 7 objectifs, on peut projeter ces résultats sous la forme d'une matrice de $7 \times 7$ graphes en deux dimensions (figure 3.5). L'observation de ces graphes révèle plusieurs motifs, confirmant nos hypothèses (figure 3.6).

Le graphe présentant les valeurs selon l'objectif 2 en fonction de l'objectif 0 (figure 3.6(a)) se divise ainsi clairement deux zones : la première, horizontale, correspondant à la zone 1 de la figure 3.1(b), regroupe les individus n'obtenant pas une fitness suffisante selon l'objectif 0 pour obtenir une fitness non-minimale selon l'objectif 1 ; la deuxième, correspondant à la zone 2 de la figure 3.1(b), présente une forme triangulaire montrant une dépendance linéaire entre les deux objectifs. Cette linéarité provient de la formulation choisie des objectifs, la fitness correspondant à l'objectif 1 incluant celle de l'objectif 0 ; en d'autres termes, plus le temps pour atteindre la lampe 0 est court, plus le temps pour atteindre la lampe 1 pourra être court. La forme en « L » du graphe correspondant à l'objectif 6 en fonction de l'objectif 0 (figure 3.6(b)) s'explique de manière similaire mais exprime une dépendance encore plus forte. Il est en effet nécessaire d'obtenir un très bon score sur la première lampe pour avoir le temps d'atteindre la dernière. Certains graphes présentent une forme symétrique curieuse, par exemple celui montrant l'objectif 4 en fonction du 1 (figure 3.6(c)), qui reflète la multiplicité des chemins pour atteindre cet objectif. Dans cet exemple, puisque la lampe 0 allume la lampe 1 et 4, il existe deux ordres possibles pour actionner l'interrupteur 4, avant et après le 1. Il est donc possible d'obtenir soit une meilleure fitness pour l'objectif 4 que pour le 1, soit l'inverse. Il est par contre impossible d'obtenir la même fitness dans les deux cas, ce qui signifierait que le robot peut atteindre simultanément deux positions. Enfin, le graphe reliant l'objectif 6 à l'objectif 1 (figure 3.6(d)) montre une grande quantité d'individus sur les axes, c'est-à-dire ayant obtenu une fitness non-minimale pour l'un des objectifs et pas pour l'autre. Cette forme s'explique par le fait qu'il n'est pas nécessaire d'atteindre la lampe 1 pour atteindre la 6. On observe donc peu de dépendances entre ces deux objectifs.

## 3.4 CONCLUSION

Dans ce chapitre, nous avons proposé la création d'un gradient de sélection pour chaque sous-tâche en utilisant les concepts de l'optimisation multiobjectif. La méthode proposée est originale et a résolu efficacement le problème avec lequel nous l'avons testée. Elle pourrait donc constituer une solution élégante et automatisée au problème du *bootstrap*, important en robotique évolutionniste. Comparé aux méthodes précédemment publiées, elle requiert moins de connaissance sur la structure du problème et contraint moins la recherche. Notamment, les sous-tâches n'ont pas besoin d'être ordonnées et différentes hypothèses peuvent être envisagées simultanément. De plus, le processus change automatiquement d'objectif tout en continuant à améliorer les solutions pour les sous-tâches. Une telle approche peut être vue comme une « exaptation dirigée » : plusieurs gradients de selection sont exploités pour mener à un comportement complexe via des exaptations mais l'expérimentateur choisit les gradients de sélection qu'il juge potentiellement pertinents.

Pour conserver un expérience simple, nous avons choisi de ne pas utiliser d'opérateur de recombinaison. L'exploitation d'un tel opérateur pourrait cependant améliorer significativement les performances de la méthode. En effet, puisque le processus maintien dans la population des spécialistes pour chaque sous-tâche, un tel opérateur pourrait réussir à combiner de tels spécialistes pour obtenir un individu capable de résoudre efficacement plusieurs sous-tâches. Isoler la partie d'un individu lui permettant de réussir une sous-tâche est cependant un problème délicat qui amène à considérer la modularité de la solution.

Plus généralement, la méthode présentée dans ce chapitre ne permet pas de faire évoluer des « briques comportementales » dont la fonction n'est pas mesurable par l'observation du comportement du robot. Par exemple, l'obtention d'un sous-réseau de neurones capable de calculer la dérivée d'un signal peut être une étape importante pour les problèmes de contrôle non-linéaires, comme le maintien vertical d'un pendule inversé (voir la section 2.5.1). On pourrait donc souhaiter l'inclure comme étape dans le processus incrémental afin de sélectionner les individus possédant la bonne « brique ». Tout comme l'utilisation d'un opérateur de croisement, il est alors nécessaire de pouvoir isoler chaque brique pour pouvoir estimer sa fitness, c'est-à-dire décomposer les solutions en modules. Des éléments de réponse pour enrichir ce point de vue seront donnés dans le chapitre 5.

# 4

# DIVERSITÉ COMPORTEMENTALE

RÉSUMÉ. Le maintien de la diversité des individus dans une population est nécessaire au bon fonctionnement des algorithmes évolutionnistes. Des études récentes montrent que l'ajout d'un objectif maximisant la distance moyenne entre les solutions candidates pouvait être bénéfique aux performances de ces algorithmes. L'application de cette idée à l'évolution de la topologie des réseaux de neurones est difficile en raison du coût computationnel du calcul de la distance entre des graphes et du nombre des réseaux possibles. Dans ce chapitre, nous proposons de maintenir la diversité des *comportements* des réseaux de neurones au lieu de celle de leur génotypes ou phénotype. Nous testons cette approche sur l'évolution de réseaux de neurones calculant une fonction logique et sur le problème du robot phototrope utilisé dans le chapitre précédent.

## 4.1 INTRODUCTION

LE PROBLÈME du *bootstrap*, abordé dans le chapitre précédent, et le problème des minima locaux ont pour origine commune l'absence de gradient de fitness exploitable pour sélectionner les individus. Toutes les variations induites par les mutations et recombinaisons mènent alors à des individus avec une fitness égale, dans le cas du *bootstrap*, ou inférieure à celle des meilleurs individus, dans le cas des minima locaux. Dans le meilleur des cas, l'évolution dans ces situations est équivalente à une recherche aléatoire car rien ne guide le processus vers des solutions particulières ; dans le pire des cas, on observe une convergence prématurée de l'algorithme vers une unique solution, souvent dégénérée. Dans les deux cas, l'optimisation ne mène généralement pas à des solutions satisfaisantes.

Afin d'éviter ces situations, il a été reconnu très tôt et confirmé expérimentalement qu'il était nécessaire de préserver la diversité dans la population. Une population répartie sur l'ensemble de l'espace de recherche a en effet plus de chances de trouver une solution qu'une autre concentrée autour d'un unique pic de fitness. L'heuristique sous-jacente suppose que si les performances ne progressent pas, il vaut mieux explorer l'espace le plus efficacement possible que concentrer les solutions autour des solutions les plus performantes. Dans ce chapitre, nous nous intéressons à la mise en place de cette idée pour l'évolution de la topologie et des poids de réseaux de neurones afin de proposer une

approche ne nécessitant pas la définition d'étapes intermédiaires, par opposition à la technique présentée dans le chapitre précédent.

Plusieurs techniques ont été proposées pour conserver une diversité importante (voir Burke et al. (2002) pour un aperçu) parmi lesquelles le *fitness sharing* (Goldberg et Richardson, 1987) est probablement la méthode la plus populaire : l'espace de recherche est modifié en diminuant la fitness des individus similaires, encourageant ainsi la recherche dans des régions encore peu explorées. Plus précisément, si $d_{ij}$ est la distance entre les *génotypes* $i$ et $j$, $\sigma_{share}$ et $\alpha$ étant des paramètres que doit choisir l'utilisateur, la *fonction de sharing* $sh(d_{ij})$ est définie de la façon suivante :

$$Sh(d) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma_{\text{share}}})^{\alpha}, & \text{si } d \leq \sigma_{\text{share}} \\ 0 & \text{sinon} \end{cases}$$

La fitness de chaque individu $i$ est alors divisée par le coefficient de niche $c_i$, qui correspond à une mesure du nombre d'individus similaires à $i$ :

$$F_i = \frac{F(x_i)}{c_i} \text{ où } c_i = \sum_{j=1}^{N} Sh(d_{ij})$$

La distance entre individus est typiquement calculée dans l'espace des génotypes ou des phénotypes, par exemple via une distance euclidienne entre les solutions. Dans certains algorithmes multiobjectif comme MOGA (Fonseca et Fleming, 1993), elle est aussi calculée dans l'espace des objectifs afin d'inciter à couvrir le plus possible le front de Pareto.

Diviser la fitness par le coefficient de niche revient à agréger deux objectifs, la fitness et la diversité. Compte tenu des avancées dans le domaine des algorithmes multiobjectifs, il semble naturel d'essayer d'éviter cette agrégation et de considérer la diversité comme un deuxième objectif à part entière. Un problème simple-objectif est alors transformé en un problème bi-objectif modélisant le classique compromis entre exploration et optimisation (Sutton et Barto, 1998). Une solution est alors non-dominée si elle est plus performante que toutes les autres ou si elle est très différente des autres.

Plusieurs travaux ont testé cette approche sur divers problèmes. En programmation génétique, de Jong et al. (2001) ont ainsi ajouté la distance moyenne aux autres membres de la population comme deuxième objectif pour les problèmes de 3,4 et 5-parité. Toffolo et Benini (2003) ont employé une approche similaire en utilisant la somme des distances euclidiennes entre un individu et tous les autres comme mesure de diversité. Ils ont aussi proposé de n'utiliser que la distance du plus proche individu. Abbass et Deb (2003) et Bui et al. (2005) ont analysé différentes variantes et les ont comparées à des optimisations avec un unique objectif. Six idées ont été envisagées pour le deuxième objectif : la génération d'apparition d'un individu (à minimiser), une valeur aléatoire, l'inversion de l'objectif, la distance au plus proche voisin (espace des génotypes), la distance moyenne à tous les individus et la distance au meilleur

individu de la population. Tous ces travaux concluent à la supériorité des approches multiobjectifs pour le maintien de la diversité. Parmi les différentes variantes envisagées, la distance au plus proche voisin (espace des génotypes) et la distance moyenne à tous les individus semblent donner les meilleurs résultats (Bui et al., 2005).

Les méthodes performantes de maintien de la diversité, en simple ou multiobjectif, reposent donc toutes sur le calcul d'une distance entre individus. Si cette approche est simple pour des génotypes de type vecteur de nombres réels ou chaîne binaire, elle est plus difficile pour les génotypes complexes comme les graphes ou les arbres de syntaxe.

Dans ce dernier cas, de Jong et al. (2001) utilisent la somme des distances entre les nœuds correspondants des deux arbres, la distance entre nœuds étant égale à 0 s'ils sont égaux et à 1 sinon. Pour les graphes, et donc les réseaux de neurones, de multiples distances accompagnées d'algorithmes de calcul ont été définies (Bunke, 2000). La plus utilisée est la distance d'édition, correspondant au nombre d'opérations nécessaires pour passer d'un graphe à l'autre, qui peut être équivalente, selon les coûts des opérations, à la taille du plus grand sous-graphe commun, une autre mesure souvent employée. Néanmoins, le calcul de cette distance est NP-complet, ce qui rend son utilisation impossible dans le cadre du maintien de la diversité car de très nombreuses distances doivent être calculées (de l'ordre du carré de la taille de la population, le plus souvent). La mesure de similarité entre graphes ayant de très nombreuses applications en *data-mining*, en bio-informatique et en reconnaissance de formes, plusieurs techniques approchées ont été publiées, comme par exemple le *graph probing* (Lopresti et Wilfong, 2003), qui compte les occurences de sous-graphes simples, ou l'utilisation de méta-heuristiques (Sorlin et Solnon, 2005). Comme nous l'avons vu dans la section 2.3.2, NEAT (Stanley et Miikkulainen, 2002a) utilise les numéros d'innovation pour résoudre le problème de la distance entre réseaux de neurones. Cette distance, qui ne peut être calculée que si tous les réseaux partagent un ancêtre commun, est ensuite notamment utilisée pour définir des niches et utiliser le *fitness sharing* de façon à donner l'opportunité à de nouvelles topologies d'être sélectionnées même si leur fitness est inférieure à celle les topologies existantes.

La complexité de l'utilisation d'une distance entre des graphes combinée à l'intérêt potentiel de l'ajout d'un objectif favorisant la diversité de la population nous a conduit à suivre une autre approche : plutôt que de maintenir une diversité entre les réseaux de neurones, nous proposons de nous intéresser à la diversité des *comportements* de ces réseaux lors de leur simulation.

## 4.2 DIVERSITÉ ET NOUVEAUTÉ COMPORTEMENTALES

Dans les algorithmes évolutionnistes classiques, la fitness récapitule le comportement d'une solution et il est supposé qu'elle contient suffisamment d'informations pour permettre de distinguer les solutions. Dans les cas les plus

simples, par exemple dans le problème du voyageur de commerce (TSP) ou dans l'optimisation de fonctions réelles, la fitness est évaluée directement à partir des propriétés du phénotype. La simulation d'un réseau de neurones et, d'une manière plus générale, de robots, ajoute une étape supplémentaire après la traduction du génotype au phénotype. Les propriétés du réseau ne permettent pas de calculer la fitness et cette dernière cache sous une unique valeur scalaire le comportement des solutions pendant une simulation qui a pu s'étendre pendant de nombreux pas de temps.

De nombreux réseaux de neurones aux génotypes et phénotypes différents peuvent avoir exactement le même comportement pour les mêmes données d'entrée. Tous les réseaux où il n'existe pas de chemin reliant les entrées aux sorties auront, par exemple, une sortie nulle. Quelle que soient leur topologie, leur nombre de neurones et le poids de leurs connections synaptiques, ces solutions auront donc la même fitness alors qu'elles auront un génotype très différent. En outre, elles sont très résistantes aux variations car la majorité des mutations et croisement ne connectera pas les entrées et sorties. Ces solutions peuvent paraître très dégénérées et donc facile à éliminer via une analyse du graphe, mais on peut en imaginer de nombreuses variations, par exemple toutes les solutions connectant directement les entrées aux sorties, le reste du réseau étant inactif, ou les réseaux aux poids synaptiques si forts que tous les neurones sont saturés quelles que soient les données d'entrée. Ces exemples soulignent qu'à la complexité du calcul de distance entre deux réseaux de neurones, rendant computationellement difficile les mesures de diversité sur le génotype ou le phénotype, s'ajoute le problème que de très nombreux génotypes peuvent mener à un même comportement. En outre, maximiser la diversité des topologies peut se faire simplement en augmentant à l'infini la taille des réseaux menant rapidement à des réseaux longs à évaluer et inutilement complexes[1]. Lors de l'évolution de réseaux de neurones, mesurer la similarité entre les réseaux semble donc peu efficace pour éviter les problèmes des minima locaux et du *bootstrap*. Il semble plus pertinent de s'intéresser à maintenir une diversité parmi les *comportements* des individus de la population.

Les résultats sur les mesures de diversité sur le génotypes (Bui et al., 2005) suggèrent de définir l'originalité $B(i)$ du comportement de l'individu $i$ comme la distance moyenne aux $N$ autres individus. Si l'on défini une distance $d(i,j)$ entre les comportements des individus $i$ et $j$, au lieu de maximiser un ensemble d'objectifs $F_1(i), F_2(i), \cdots, F_n(i)$, nous sommes amenés à résoudre le problème multiobjectif :

$$Maximiser : \begin{cases} F_1(i) \\ \vdots \\ F_n(i) \\ B(i) = \frac{1}{N} \sum_{j=1}^{N} d(i,j) \end{cases}$$

1 Il est toutefois possible d'utiliser un troisième objectif poussant à réduire la taille des réseaux, comme le font (de Jong et al., 2001) pour la programmation génétique

Mettre au point une distance entre comportements peut sembler particulièrement difficile mais les tests préliminaires que nous avons conduits et les résultats de la section suivante montrent que dans beaucoup de cas des distances très simples sont suffisantes pour éviter la plupart des minima locaux et donc changer fondamentalement les performances des algorithmes. Une méthode simple pour les robots mobiles est d'associer à chaque individu $i$ un vecteur $\mathbf{v}^{(i)}$ décrivant l'état de l'environnement à la fin de l'expérience. Par exemple, si un robot doit bouger des objets dans une arène, $\mathbf{v}^{(i)}$ peut contenir la position des objets ; s'il doit explorer un labyrinthe, sa position finale peut aussi être utilisée ; s'il doit atteindre des zones, $\mathbf{v}_k^{(i)}$ pourra prendre la valeur 1 si la zone $k$ est atteinte et 0 sinon, etc. La distance $d(i,j)$ peut ensuite être facilement définie comme la distance euclidienne entre $\mathbf{v}^{(i)}$ et $\mathbf{v}^{(j)}$ :

$$d(i,j) = \left|\left|\mathbf{v}^{(j)} - \mathbf{v}^{(i)}\right|\right|$$

N'importe quelle autre distance peut bien sûr être utilisée mais tous les MOEA ne peuvent pas être employés car la valeur de $B(i)$ dépend de la population courante. Par conséquent, elle variera à chaque génération et un individu non-dominé grâce à une valeur élevée de $B(i)$ pourra être dominé à la génération suivante. Le MOEA doit donc recalculer le classement de tous les individus à chaque génération. Il est donc par exemple possible d'utiliser les algorithmes MOGA et NSGA-II mais pas $\varepsilon$-MOEA.

Dans des travaux parallèles à ceux-ci mais portant sur l'ouverture (*open-endness*) de l'évolution, Lehman et Stanley (2008) ont proposé de maximiser la nouveauté *à la place* de la fitness. La nouveauté est calculée à l'aide d'une distance entre comportements similaires à celle que nous venons de définir. Contrairement à la diversité, calculée en utilisant uniquement la population courante, la nouveauté prend en compte l'ensemble des comportements déjà rencontrés en utilisant une archive. Plus précisément, Lehman et Stanley mesurent la nouveauté $\rho$ d'un individu $x$ en calculant la distance moyenne à ses $k$ plus proches voisins :

$$\rho(x) = \frac{1}{k}\sum_{i=0}^{k} d(x,\mu_i)$$

où $k$ est un paramètre défini expérimentalement, et $\mu_i$ est le ième plus proche voisin de $x$ en fonction de la distance $d(i,j)$ et de tous les individus rencontrés depuis le début du processus évolutionniste.

Bien que les auteurs proposent de remplacer la fitness par $\rho$, il est possible de l'utiliser comme deuxième objectif de la même manière que pour l'objectif de diversité. On obtient alors une nouvelle méthode d'exploration, distincte de la diversité comportementale et de la diversité de Lehman et Stanley (2008). De plus, en prenant en compte à la fois l'archive et la population courante, ces derniers mélangent les effets de la recherche de nouveaux comportements avec le maintien de la diversité. Dans une seconde série d'expériences, nous proposons donc d'utiliser la distance des plus proches membres de l'archive

comme deuxième objectif, cherchant ainsi à récompenser de nouveaux comportements. Malgré leurs similitudes, trois différences entre nouveauté et diversité comportementales peuvent s'avérer importantes :

- le fait de ne pas utiliser la population courante mais une archive pour calculer la nouveauté peut beaucoup changer la dynamique du processus évolutionniste ;
- la diversité peut mener à des cycles de comportements – un comportement peut redevenir périodiquement sélectionné car différent du reste de la population – alors que la nouveauté évite les cycles grâce à son historique ;
- le coût computationnel de la prise en compte d'une archive peut vite s'avérer prohibitif, suivant la distance et la structure de données choisie.

Afin de pouvoir comparer la recherche de nouveauté et celle de diversité avec une technique basée sur le phénotype (la topologie du réseau de neurones), nous avons implémenté la technique de *graph probing* décrite par Lopresti et Wilfong (2003) pour calculer une approximation de la distance entre deux graphes. Cette distance très simple a déjà été utilisée avec un algorithme évolutionniste pour la reconnaissance de symboles (Reveaux et al., 2007). Soient $G$ un graphe comportant $N$ arcs et $k_{ij}$ le nombre de sommets de $G$ comportant $i$ arcs entrants et $j$ arcs sortants, on défini le vecteur d'entiers $\mathbf{v}^{(G)}$ de manière à regrouper les différentes valeurs de $k$ :

$$\mathbf{v}^{(G)}_{i\times N+j} = k_{ij}$$

La distance $d_G$ entre les graphes $G_1$ et $G_2$ peut alors être approchée en réalisant la distance euclidienne entre $\mathbf{v}^{(G_1)}$ et $\mathbf{v}^{(G_2)}$ :

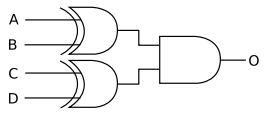
$$d_G(G_1, G_2) = \left|\left|\mathbf{v}^{(G_1)} - \mathbf{v}^{(G_2)}\right|\right|$$

## 4.3 EXPÉRIENCES

### 4.3.1 *Fonction logique :* $\left[(a \oplus b) \wedge (c \oplus d)\right]$

*Contexte expérimental*

Nos premiers tests ont porté sur la fonction logique $\left[(a \oplus b) \wedge (c \oplus d)\right]$ où $a, b, c$ et $d$ sont des booléens et $\oplus$ le ou exclusif (xor). Cette fonction est raisonnablement complexe mais, surtout, très déceptive. En effet, l'analyse de sa table de vérité (figure 4.1) montre qu'un réseau renvoyant toujours faux aura la bonne réponse dans 75% des tests (12 cas sur 16). On peut donc s'attendre à ce que des réseaux de neurones très simples dominent rapidement la population, empêchant l'émergence de bonnes solutions. En utilisant un algorithme élitiste, une population de 1000 individus et un codage direct de la topologie d'un réseau de portes NAND, Kashtan et Alon (2005) rapportent que seules 36 expériences sur 50 ont trouvé des solutions en moins de $10^5$ générations.




| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

FIG. 4.1.: Logigramme et table de vérité de la fonction $\big[(a \oplus b) \wedge (c \oplus d)\big]$ où $a, b, c$ et $d$ sont des booléens et $\oplus$ le ou exclusif (xor).

La fitness est simplement définie comme la somme des erreurs pour chaque jeu d'entrées :

$$F_x = 1 - \frac{1}{16} \sum_{i=1}^{16} |o_i - d_i|$$

où $o_i$ est la sortie du réseau de neurones pour le jeu d'entrée $i$ et $d_i$ la sortie attendue. Chaque réseau est simulé pendant 100 pas de temps et on attribue une fitness arbitrairement basse à ceux dont la sortie n'est pas constante pendant les 10 derniers pas de temps.

Afin d'utiliser l'objectif de diversité comportementale, on associe à chaque individu $x$ un vecteur de booléens $\mathbf{v}^{(x)}$ contenant la sortie du réseau pour chacun des 16 jeux d'entrées différents :

$$\mathbf{v}_i^{(x)} = nn_x(b_i), i \in 1, 2, \cdots, 16; b_i \in 0000, 0001, \cdots, 1111$$

où $nn_x(b_i)$ désigne la sortie du réseau de neurone correspondant à $x$ pour l'entrée $b_i$. La distance entre les comportements de $x_1$ et $x_2$ est définie comme la distance euclidienne entre les vecteurs $\mathbf{v}^{(x_1)}$ et$\mathbf{v}^{(x_2)}$. Dans un contexte sensiblement différent, García-Pedrajas et al. (2002) et Moriarty (1998) ont mis en place une idée similaire pour préserver la diversité fonctionnelle entre les modules dans leurs systèmes de co-évolution coopérative : chacun d'eux est soumis à un jeu d'entrées particulier et leurs sorties sont notées dans un vecteur utilisé ensuite pour calculer une distance.

Pour implémenter la nouveauté comportementale, une archive $A_n$ peut-être construite en stockant les vecteurs $\mathbf{v}^{(x)}$ rencontrés parmi les $2^{16} = 65536$ combinaisons possibles :

$$A_n(\mathbf{v}) = \begin{cases} 1 \text{si } \mathbf{v} \in P_1, P_2, \cdots, P_n \\ 0 \text{ sinon} \end{cases}$$

où $P_n$ désigne les comportements de la population à la génération $n$. Le score de nouveauté d'un individu est alors égal à la distance euclidienne moyenne aux

$k = 10$ plus proches individus de l'archive, $k$ étant un chiffre arbitraire choisi en fonction de nos expériences et des résultats de Lehman et Stanley (2008).

Le coût computationnel ajouté par cette archive est important en raison de sa taille, plus importante que celle de la population. Pour une population de taille $N$, au lieu d'effectuer au maximum $N^2$ calculs de distance entre individus comme dans le cas de la diversité, $N \times (2^{16})$ calculs peuvent être requis. Cependant, l'utilisation d'une structure adaptée de type KD-tree (Bentley, 1975) peut permettre de réduire significativement la complexité du calcul de la distance entre un individu et ses plus proches voisins dans l'archive.

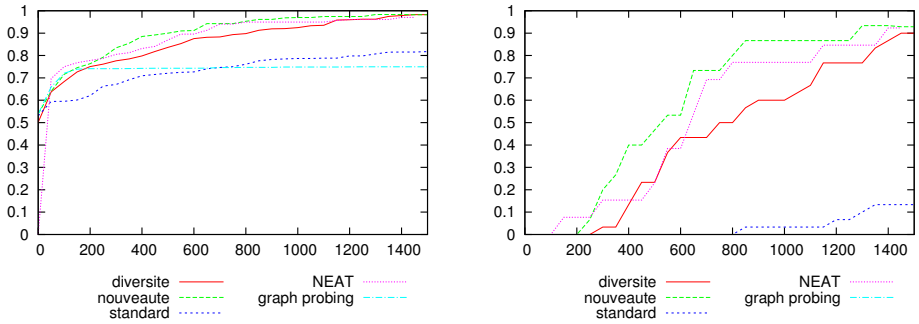
Un codage direct sans contraintes sur la topologie dont les paramètres sont décrits dans l'annexe A.2.2 a été utilisé avec l'algorithme NSGA-II et une population de 400 individus. Le croisement n'est pas utilisé. Au total, 5 séries de 16 expériences ont été lancées :

1. simple-objectif en utilisant uniquement la fitness ;
2. deux objectifs, avec un objectif de diversité comportementale ;
3. deux objectifs avec un objectif de nouveauté comportementale (avec une archive) ;
4. deux objectifs avec un objectif de diversité phénotypique basé sur le *graph probing* ;
5. simple-objectif en utilisant NEAT, qui utilise un maintien de la diversité basé sur le phénotype et les numéros d'innovation ainsi qu'une évolution « incrémentale » où tous les réseaux de la première génération ont une topologie identique, toutes les entrées étant reliées aux sorties ; la possibilité de créer des connexions récurrentes n'est pas activée.

*Résultats*

La figure 4.2(a) montre l'évolution de la moyenne des meilleures fitness pour les 16 expériences dans les 5 cas étudiés et la figure 4.2(b) la proportion d'expériences ayant convergé à chaque génération, c'est-à-dire où la meilleure fitness dépasse 0.95. Les meilleures performances sont atteintes pour les deux méthodes basées sur les comportements (diversité et nouveauté) et par NEAT, qui devancent largement l'approche simple-objectif et l'approche utilisant le *graph probing*. Celle basée sur l'archive mène à des fitness légèrement meilleures que les autres pendant les premières générations mais cette différence est peu significative puiqu'il faut aux trois meilleures approches un nombre de générations similaires (environ 1300) pour que 90% des expériences convergent. Si peu d'expériences (12%) en simple objectif ont convergé en 1500 générations, en 30000 génération, cette méthode converge dans 86% des cas.

Afin de mieux analyser la différence statistique entre NEAT, diversité comportementale et nouveauté comportemantale, on peut s'intéresser au nombre moyen de générations requis pour converger ($f > 0.95$) et utiliser un T-test de Student. On obtient les résultats de la table 4.1. Dans les trois cas, les différences sont peu voire pas significatives, la génération moyenne de conver-




(a) (b)

FIG. 4.2.: (a) Fitness moyenne en fonction de la génération pour la fonction $\left[(a \oplus b) \wedge (c \oplus d)\right]$, pour 30 expériences. (b) Proportion d'expériences ayant convergé (fitness supérieure à 0.95) à chaque génération. Aucune expérience n'a convergé dans l'expérience utilisant le graph probing en 1500 générations mais certaines ont convergé après. De même, 86% des expériences en simple-objectif ont convergé en 30000 générations.

| | Diversité | Nouveauté | NEAT |
|---|---|---|---|
| Gén. moyenne | 921.7 | 734.6 | 757.3 |
| Écart type | 463.1 | 494.7 | 499.4 |
| T-test Div. | p=1.0 | p=0.2401 | p=0.3025 |
| T-test Nouv | p=0.2401 | p=1.0 | p=0.9083 |
| T-test NEAT | p=0.3025 | p=0.9083 | p=1.0 |

TAB. 4.1.: Génération moyenne de convergence et T-test de Student

.

gence se situant entre 700 et 900 génération avec un écart type de l'ordre de 500 générations.

Ajouter un objectif de diversité ou de nouveauté des comportements améliore donc très significativement les performances de l'évolution au point d'être équivalentes à celles obtenues avec NEAT. Cette équivalence est d'autant plus étonnante que les méthodes proposées sont beaucoup plus simples que NEAT (une simple distance moyenne entre comportements contre des mécanismes d'espèces, du fitness sharing et des numéros d'innovation) et que NEAT réduit de manière très significative la taille de l'espace de recherche en commençant l'évolution avec une unique topologie, assez proche des topologies capables de résoudre le problème. De plus, dans cette expérience, les connexions récurrentes sont interdites dans NEAT alors qu'elles sont possibles dans le codage direct utilisé, ce qui change la taille de l'espace de recherche. L'ajout de l'archive ne semble pas influer beaucoup sur le résultat de l'expérience malgré un coût computationnel important. Il est cependant difficile de généraliser cette conclusion sans avoir conduit des expériences sur d'autres problèmes.

La technique de maintien de la diversité par une distance de graphe, par contre, n'a pas amélioré sensiblement les résultats. Plusieurs explications sont possibles : la distance de graphe n'est peut-être pas assez précise, étant donné qu'il ne s'agit que d'une approximation de la distance d'édition ; il peut aussi être indispensable de ne calculer la distance que sur les sous-réseaux connectés aux entrées et aux sorties afin de ne rendre compte que de la partie utile du réseau ; enfin, du fait de leur grande influence sur le comportement du réseau, les poids nécessitent peut-être d'être pris en compte. Résoudre ces problèmes nécessiterait la mise en place de méthodes beaucoup plus complexes que celles mise en place pour le comportement.

Afin de mieux comprendre la dynamique induite par l'ajout des nouveaux objectifs, nous avons mis en correspondance la fitness avec la proportion d'individus de la population ayant la réponse « vrai » pour chacun des 16 jeux d'entrée, à chaque génération (Figures 4.3, 4.4, 4.5, 4.6). Si cette analyse ne permet pas de connaître le nombre d'individus ayant chacun des $2^{16}$ comportements possibles, elle permet de visualiser l'évolution de la diversité des comportements. Les motifs observés dans chacun des cas sont étonnamment différents.

Dans le cas de l'évolution en simple-objectif (figure 4.3), la population apparaît très uniforme, 90 à 100% des individus renvoyant faux dans tous les cas et donnant donc une réponse fausse dans 4 cas. Au bout de 400 générations, une partie de la population commence à donner la bonne réponse dans un cas supplémentaire et en moins de 100 générations l'ensemble de la population acquiert ce comportement. Cette figure confirme l'intuition des difficultés liés à la fonction $\left[(a \oplus b) \wedge (c \oplus d)\right]$ et la nécessité de mieux maintenir la diversité. L'approche maintenant la diversité du phénotype via la distance de graphe (figure 4.4) semble maintenir une certaine diversité de comportements car aucun comportement ne semble être partagé par 100% de la population. Cependant, peu d'individus renvoient vrai dans les 4 cas nécessaires, barrant la figure de 4

bandes grises et, malgré cette apparente diversité, les fitness observées ne sont pas meilleures que celles observées en simple-objectif. L'objectif de diversité comportementale (figure 4.5) semble, lui, maintenir une diversité presque parfaite, la figure se présentant comme presque uniformément grise. L'alternance rapide des niveaux de gris rend compte de cycles courts où les comportements sont successivement sélectionnés puis défavorisés par l'objectif de diversité.

En dépit de performances similaires voire légèrement meilleures qu'en utilisant l'objectif de diversité, l'approche utilisant une archive (figure 4.6) montre une dynamique très différente où les moments de faibles et fortes diversités alternent. La taille de l'archive augmente, comme on peut s'y attendre quand la diversité des comportements augmente mais reste parfois presque stable pendant plusieurs centaines de générations. Afin d'expliquer ce comportement, il faut tout d'abord noter que l'archive telle que nous l'avons implémentée possède malgré sa formulation simple une propriété peu intuitive : un individu ayant un comportement nouveau à une génération sera mécaniquement beaucoup moins nouveau à la génération suivante car son comportement sera présent dans l'archive. Par conséquent, l'évolution ne sélectionnera cet individu que tant qu'il aura peu de voisins dans l'archive, c'est-à-dire dans notre cas des variantes où seules quelques composantes du vecteur de comportement changent. Lorsque toutes ces variantes ont été explorées, rien ne pousse l'algorithme à maintenir une diversité dans sa population ni à sélectionner les comportements les moins rencontrés ; il convergera alors vers un petit nombre de comportements avec une bonne fitness, la taille de l'archive restant presque constante. Lorsqu'un nouveau comportement finit par apparaître suite à une mutation, la pression sur la nouveauté fait qu'il explore rapidement tous les comportements voisins afin de remplir l'archive. Ultimement, quand l'archive grossit, la pression sur la nouveauté se fait de plus en plus faible car tous les individus ont une distance équivalente aux individus de l'archive.

### 4.3.2 *Robot phototrope*

*Contexte expérimental*

Nous avons déjà plusieurs fois évoqué le fait que le problème du *bootstrap* était un cas particulier du problème des minima locaux. La technique de maintien de la diversité des comportements présentée se montrant efficace pour éviter les difficultés dues aux minima locaux, on peut aussi la tester sur un problème incrémental. Dans les deux cas, il s'agit en effet d'explorer efficacement les différents comportements possibles en l'absence de pression de sélection par la fitness. Afin de comparer l'approche présentée dans ce chapitre avec celle du chapitre 3, nous analysons dans cette section l'application de la diversité comportementale au problème incrémental du robot phototrope décrit dans la section 3.3.

La fitness est définie comme le nombre de pas de temps requis pour atteindre la dernière lampe, c'est-à-dire le septième objectif utilisé dans la sec-

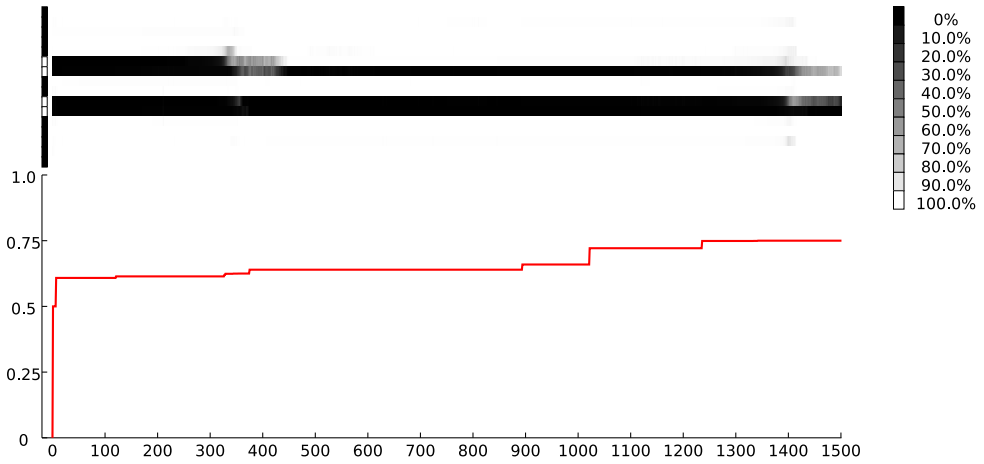



FIG. 4.3.: $\left[(a \oplus b) \wedge (c \oplus d)\right]$ : analyse des comportements des individus pendant l'évolution simple-objectif dans un expérience typique. (haut) Proportion la population ayant la sortie correcte pour chacun des 16 jeux d'entrées, en fonction de la génération. Les rectangles noir et blanc à l'extrémité gauche de la figure représentent le résultat attendu (blanc pour vrai, noir pour faux). On peut noter que dès la première génération, la majorité des individus obtiennent la bonne réponse dans 12 cas sur 16. (bas) Fitness du meilleur individu en fonction de la génération.

tion 3.3. Elle correspond à la fitness « intuitive » que nous aimerions pouvoir utiliser pour résoudre un tel problème. Trois expériences sont prises en compte pour éviter le sur-apprentissage :

$$F = F_6 = \min_{n=1,2,3} \frac{-\varphi(n,i)}{T}$$

Le comportement d'un individu $i$ est décrit par un vecteur $\mathbf{v}^{(i)}$ de booléens :

$$\mathbf{v}_k^{(i)} = \begin{cases} 1 & \text{si la k-ieme lampe a été atteinte} \\ 0 & \text{sinon} \end{cases}$$

En utilisant l'objectif de diversité, on s'attend à ce que, lorsqu'un contrôleur parvient à atteindre une nouvelle lampe, il soit non-dominé et donc sélectionné pour construire les solutions suivantes permettant de toucher l'interrupteur suivant.

L'algorithme NSGA-II a été utilisé avec une population de 200 individus. 30 expériences de 1000 générations ont été lancées. Comme dans la section 3.3, le robot est contrôlé par un simple perceptron multicouche dont les poids sont soumis à évolution par mutation gaussienne.

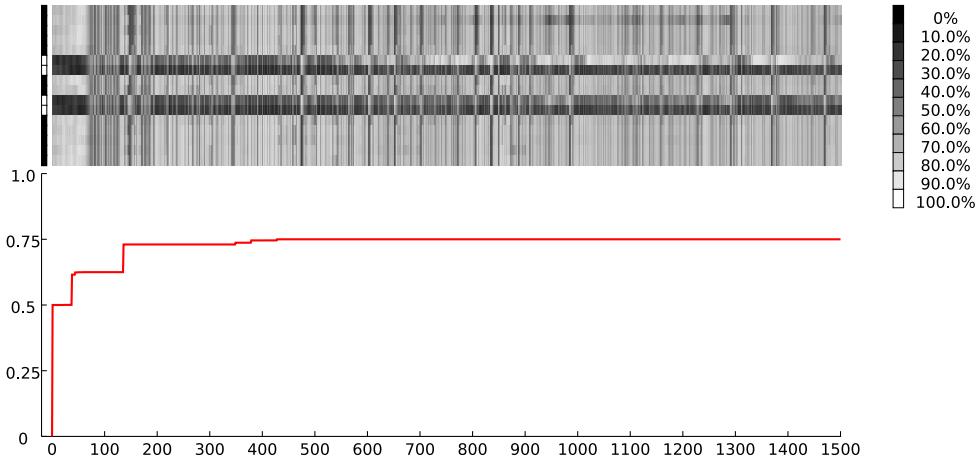



FIG. 4.4.: $\left[(a \oplus b) \wedge (c \oplus d)\right]$ : analyse des comportements des individus pendant une expérience utilisant le maintien de la diversité phénotypique par une estimation de la distance basée sur le « graph probing ». (haut) Proportion la population ayant la sortie correcte pour chacun des 16 jeux d'entrées, en fonction de la génération. (bas) Fitness principale du meilleur individu en fonction de la génération.

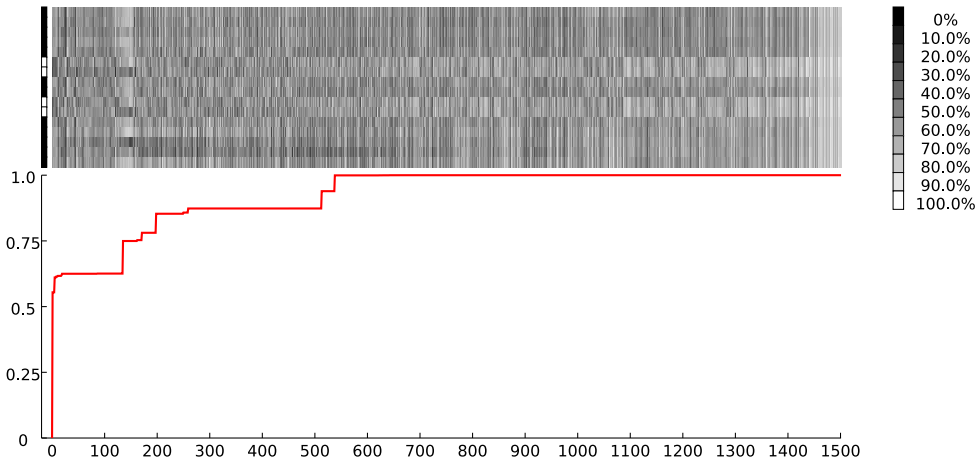



FIG. 4.5.: $\left[(a \oplus b) \wedge (c \oplus d)\right]$ : analyse des comportements des individus pendant une expérience utilisant l'objectif de diversité comportementale. (haut) Proportion la population ayant la sortie correcte pour chacun des 16 jeux d'entrées, en fonction de la génération. (bas) Fitness principale du meilleur individu en fonction de la génération.

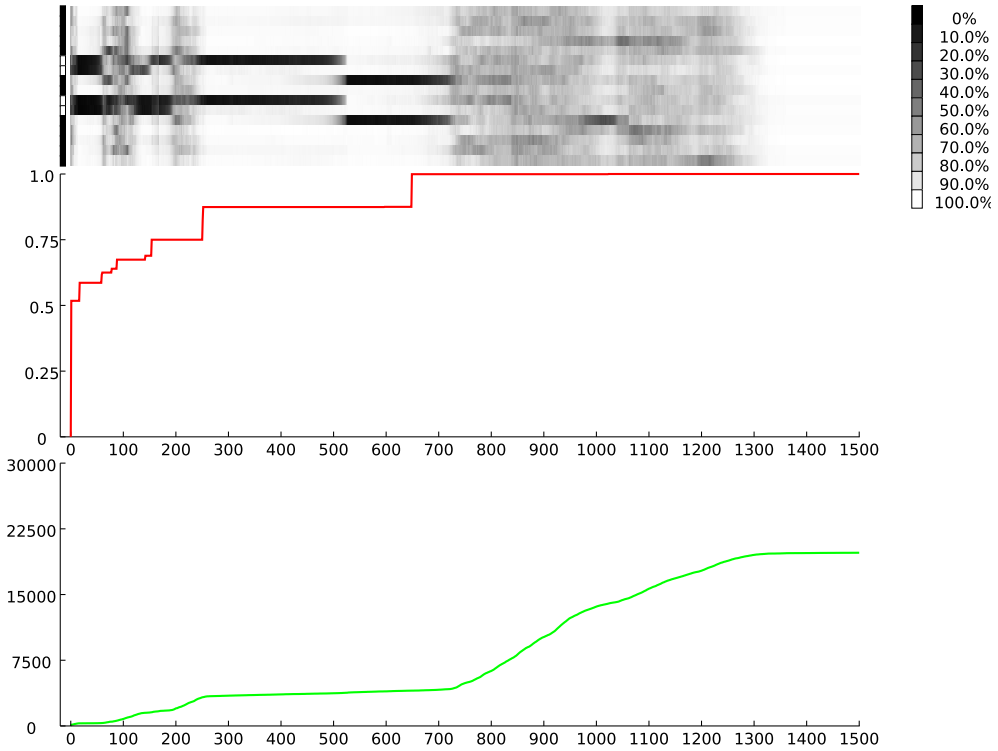



FIG. 4.6.: $\left[(a \oplus b) \wedge (c \oplus d)\right]$ : analyse des comportements des individus pendant une expérience utilisant la nouveauté comportementale. (haut) Proportion la population ayant la sortie correcte pour chacun des 16 jeux d'entrées, en fonction de la génération. (milieu) Fitness principale du meilleur individu en fonction de la génération. (bas) Taille de l'archive en fonction de la génération.

*Nouveauté comportementale*

Des expériences préliminaires ont montré que l'objectif de nouveauté présenté dans la section précédente ne peut pas être utilisé dans cette situation. À la fin de la génération aléatoire l'archive contient généralement les comportements suivants, contraints par la forme du problème :

0000000
1000000
1000100
1100000

Dans le premier cas, aucune lampe n'a été atteinte, dans le second, le premier interrupteur a été actionné, etc. Si l'on cherche à sélectionner des individus pour la reproduction, seul l'objectif de nouveauté est utilisé étant donné que les individus ont tous la même note selon l'autre objectif. Or, si l'on calcule la valeur correspondante, elle est maximum pour trois comportements sur les quatre possibles, dont le comportement n'atteignant aucune lampe :

$$\begin{aligned} 0000000 &\rightarrow \quad 0.957106781187 \\ 1000000 &\rightarrow \quad 0.75 \\ 1000100 &\rightarrow \quad 0.957106781187 \\ 1100000 &\rightarrow \quad 0.957106781187 \end{aligned}$$

Le comportement 0000000 est le plus simple à obtenir et le plus robuste aux mutations. Les autres n'amenant aucun avantage sélectif, le processus reste à nouveau bloqué à cause d'un problème de *bootstrap*. Cet effet est amplifié par la sélection par tournoi de NSGA-II car la majorité de la population des premières générations a le comportement 0000000. À l'inverse, l'objectif de diversité prend en compte implicitement la difficulté des comportements en favorisant ceux qui sont peu représentés dans la population, généralement les plus difficiles à atteindre.

*Résultats et comparaison*

Les 30 expériences lancées ont obtenu des contrôleurs résolvant la tâche demandée en moins de 300 générations avec des fitness comparables à celles obtenues par évolution incrémentale multiobjectif. La table 4.2 présente la génération moyenne de *bootstrap*, c'est-à-dire celle où la fitness principale obtient pour la première fois une valeur non-minimale. La génération de *bootstrap* est significativement plus faible avec l'objectif de diversité qu'avec l'évolution incrémentale multiobjectif (158 contre 215) mais les ordres de grandeurs sont identiques : dans les deux cas le problème du bootsrap a été surmonté en

| | Évolution incrémentale | Diversité | Nouveauté |
|---|---|---|---|
| Bootstrap | 215 (16) | 158 (11) | X |
| Fitness principale | −0.36 (0.02) | −0.34 (0.01) | X |

TAB. 4.2.: Génération moyenne de *bootstrap* et écart type ; fitness moyenne obtenue après 500 générations, pour 30 expériences. Les écarts types sont entre parenthèses.

quelques centaines de générations. Cette différence de génération provient vraisemblablement des mauvaises performances des algorithmes multiobjectif avec une fitness ayant beaucoup d'objectifs.

La figure 4.7 met en correspondance la proportion d'individus atteignant chaque lampe avec la fitness maximum. On y distingue clairement les dépendances entre les sous-tâches sous forme d'escaliers de niveau de gris. Ainsi, peu d'individus atteignent la lampe 4 pendant les 50 premières générations mais la proportion des individus y arrivant augmente progressivement. La lampe 5 présente le même motif mais sur 200 générations, soulignant la nécessité d'atteindre la lampe 4 pour pouvoir actionner l'interrupteur de la 5. La lampe 6, la dernière, forme la dernière marche de l'escalier. Un motif très similaire peut être observé dans l'enchaînement des lampes 1, 2 et 3. Comme dans les expériences du chapitre 3, la lampe 3 est atteinte bien après la lampe 6. Néanmoins, la régularité de ces dégradés cède la place à des cycles pour chacune des sous-tâches[- quand environ 50% de la population réalise l'objectif. La proportion tombe alors en une seule génération à environ 25%, puis remonte progressivement jusqu'au prochain cycle. La durée des cycles est variable mais réduit sensiblement quand le nombre de générations augmente. Lorsqu'un comportement est partagé par plus de la moitié de la population, ces individus sont par définition très proches et leur score de diversité est donc faible. La rapidité de changement peut s'expliquer par un changement de statut dans le classement des individus, par exemple un passage d'un statut de non-dominé à dominé, ce qui change les parents utilisés pour créer la génération suivante.

Ces cycles contrastent avec la régularité observée sur la figure équivalente pour l'évolution incrémentale multiobjectif (figure 4.8). Si on observe les mêmes figures en escaliers, les dégradés sont très réguliers, montrant que la population augmente progressivement le nombre d'individus capables de résoudre chaque sous-tâche mais ne le décroît jamais brusquement. Contrairement à l'expérience de la figure 4.7, on peut aussi remarquer que, en quelques générations, 100% de la population atteint la première lampe, requise pour atteindre toutes les autres. Dans le cas du maintien de la diversité, il est pertinent de conserver des individus n'actionnant pas le premier interrupteur car ils sont très différents du reste de la population. Dans le cadre de l'évolution incrémen-

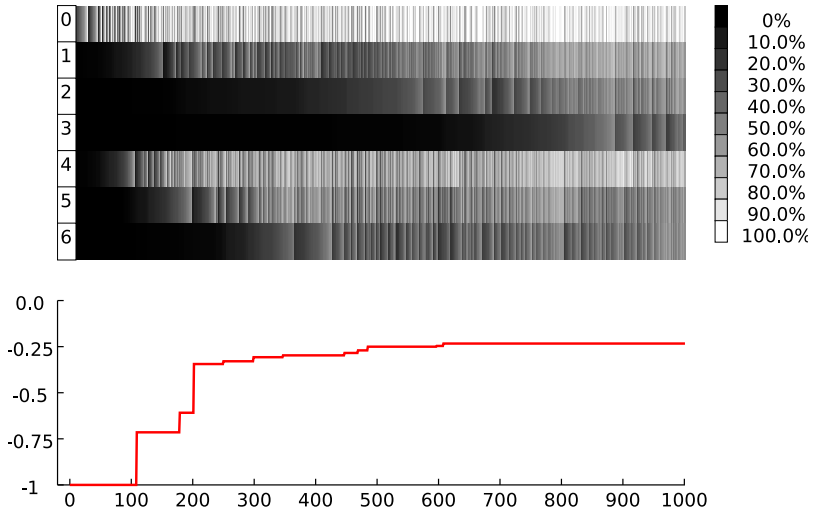



FIG. 4.7.: Robot phototrope : analyse des comportements des individus pendant une expérience utilisant la diversité comportementale. (haut) Proportion de la population atteignant chaque lampe en fonction de la génération. (bas) Fitness principale du meilleur individu en fonction de la génération. Le *bootstrap* du processus a lieu après environ 100 générations.

tale multiobjectif, ces individus sont dominés et donc progressivement éliminés de la population.

Malgré des idées de départ très différentes, les similarités des performances et des figures 4.7 et 4.8 suggèrent que les deux méthodes pourraient fonctionner de manière assez proche. En effet, elles appliquent toutes deux une pression de sélection vers l'exploration afin de favoriser les individus capables d'actionner un interrupteur que le reste de la population n'arrive pas à atteindre. Les individus sont décrits dans les deux cas par un vecteur de 7 composantes, une par lampe ; la différence entre les deux approche réside donc uniquement sur la manière dont ce vecteur est utilisé pour classer les individus[2] : un classement de Pareto pour l'évolution incrémentale et une distance moyenne à la population pour la diversité comportementale. Par conséquent, comme en témoignent les figures 4.7 et 4.8, la diversité comportementale sélectionnera les individus qui allument moins de lampes que le reste de la population, par exemple ceux n'atteignant aucun interrupteur, alors que ces individus seront dominés et donc non sélectionnés par l'évolution incrémentale. Le maintien de la diversité « gaspille » donc une part de la population à maintenir des

2 La similarité pourrait être renforcée si des valeurs binaires étaient utilisées pour l'évolution incrémentale ou si des valeurs réelles étaient utilisées pour le maintien de la diversité. De telles modifications sont compatibles avec les méthodes présentées.

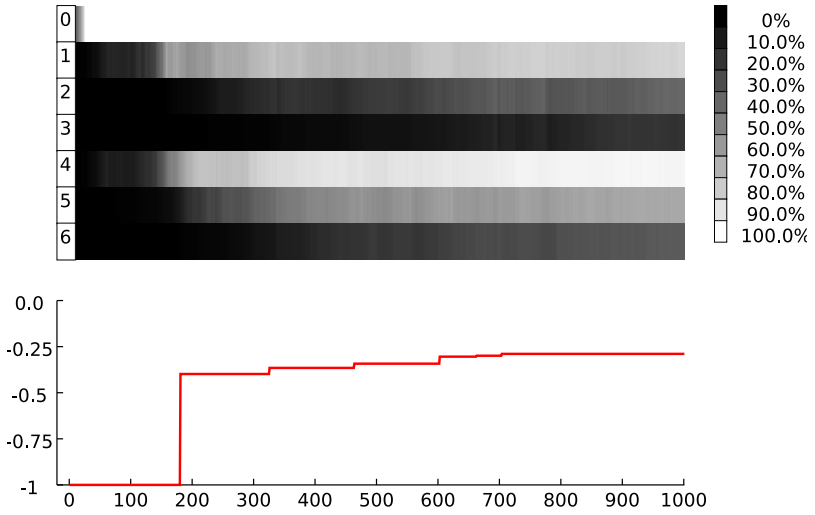



FIG. 4.8.: Robot phototrope : analyse des comportements des individus pendant une expérience utilisant l'évolution incrémentale multiobjectif. (haut) Proportion de la population atteignant chaque lampe en fonction de la génération. (bas) Fitness principale du meilleur individu en fonction de la génération. Le *bootstrap* du processus a lieu après environ 180 générations.

solutions sous-optimales alors que le classement de Pareto prend en compte qu'il vaut mieux actionner le maximum d'interrupteur. Écrit autrement, le classement de Pareto *oriente* la recherche contrairement à la distance moyenne euclidienne utilisée dans l'autre méthode. Plus généralement, les deux méthodes constituent les variantes orientée et non-orientée d'un même processus de recherche favorisant l'exploration.

## 4.4 CONCLUSION

Le maintien de la diversité dans la population est primordial pour le fonctionnement d'un algorithme évolutionniste. Plusieurs travaux récents montrent qu'ajouter un objectif récompensant les individus les plus différents du reste de la population permet de conserver une diversité importante et d'aboutir à de meilleures performances (de Jong et al., 2001; Toffolo et Benini, 2003; Abbass et Deb, 2003; Bui et al., 2005). Ces méthodes s'avèrent plus efficaces que les méthodes de *fitness sharing* (Goldberg et Richardson, 1987) souvent utilisées dans les algorithmes évolutionnistes.

Ces travaux s'appuient sur des distances entre génotypes ou entre phénotypes mais cette méthode est difficilement applicable à l'évolution de la topologie de réseaux de neurones car la distance entre génotypes dépend beaucoup

du codage, souvent complexe et celle entre phénotypes de la distance entre deux graphes, un problème NP-complet. De plus, une infinité de réseaux de neurones peuvent avoir une même sortie ou faire effectuer à un robot une même action. Il apparaît donc plus pertinent d'appliquer des techniques de maintien de la diversité multiobjectif au *comportement* du réseau de neurones ou du robot. Nous avons envisagé dans ce travail deux approches : maintenir la *diversité comportementale* en maximisant la distance moyenne entre individus et maximiser la *nouveauté comportementale* en maximisant la distance avec les comportements déjà rencontrés via l'utilisation d'une archive. Dans les deux cas, un codage direct de réseaux de neurones est utilisé, les méthodes ne dépendant pas du tout du codage.

Ces deux approches ont été comparées à NEAT, à un algorithme évolutionniste mono-objectif et à une méthode basée sur une approximation de la distance entre les phénotypes pour trouver des réseaux de neurones capables de calculer la sortie de la fonction booléenne $\left[(a \oplus b) \wedge (c \oplus d)\right]$. Les résultats obtenus démontrent l'intérêt des deux méthodes envisagées : 90% des expériences ont convergé en moins de 1500 générations, une performance équivalente à celle de NEAT, alors que moins 10% ont atteint l'objectif dans les deux derniers cas. Cette performance est d'autant plus intéressante que les méthodes sont significativement plus simples que NEAT et, contrairement à ce dernier, n'emploient aucun biais vers les réseaux *feed-forward*. L'approche basée sur l'archive ne semble pas améliorer substantiellement les performances par rapport à l'approche utilisant uniquement sur la diversité de la population, en dépit d'un coût computationnel important.

Nous avons ensuite comparé le maintien de la diversité comportementale avec l'évolution incrémentale multiobjectif présentée dans le chapitre 3 sur le problème du robot phototrope. La nouveauté comportementale ne peut être appliquée à cette tâche car elle ne prend pas suffisamment en compte la difficulté pour obtenir chaque comportement. Les résultats montrent que la méthode de la diversité comportementale permet de résoudre le problème du *bootstrap* pour la tâche étudiée dans toutes les expériences que nous avons lancées. De plus, elle requiert moins de générations que l'évolution incrémentale multiobjectif présentée précédemment. L'analyse des deux méthodes et des résultats montre qu'elles sont fondamentalement proches et exploitent presque la même information. Toutes deux constituent des variantes d'une recherche favorisant l'exploration grâce à l'optimisation multiobjectif et à l'ajout de gradients de sélection ; l'une oriente la recherche en considérant certains comportements comme plus intéressants que d'autres, l'autre pousse seulement à explorer tous les comportements possibles.

Les résultats des deux expériences que nous avons menées confirment donc l'intérêt d'ajouter un objectif de diversité comportementale à la fitness, les performances étant nettement améliorées. Dans les deux problèmes envisagés, nous avons pu réduire le comportement du réseau de neurones ou du robot à un vecteur de taille raisonnable, permettant de définir la distance entre com-

portements comme une distance euclidienne. Une telle réduction n'est cependant pas aussi triviale dans toutes les tâches envisageables. Afin d'appliquer l'approche présentée dans ce chapitre à un plus grand nombre de problèmes, il sera donc probablement nécessaire de développer un cadre théorique permettant de comparer de manière plus générique les comportements d'agents.

# 5

# ÉVOLUTION INCRÉMENTALE MODULAIRE

RÉSUMÉ. L'exaptation requiert l'apparition de modules reliés à des gradients de sélection. Dans le chapitre 3, nous avons montré que les algorithmes évolutionnistes multiobjectifs pouvaient créer de tels gradients ; dans celui-ci, nous ajoutons la notion de module. Après avoir montré que la modularité en tant que telle ne facilitait pas l'évolution, nous présentons un codage modulaire simple inspiré des concepts de parcellation et d'intégration, issus de la littérature biologique. Nous appliquons ensuite cette méthode à l'évolution de la topologie et des paramètres de réseaux de neurones pour approcher une fonction logique à la structure modulaire. Les performances sont analysées au regard de l'apport de la modularité du génotype et de l'ajout de gradients de sélection.

## 5.1 INTRODUCTION

NOUS AVONS MONTRÉ dans le chapitre 3 que l'évolution incrémentale pouvait être vue comme une optimisation multiobjectif, chaque sous-tâche correspondant a un objectif. Utiliser un algorithme multiobjectif permet ensuite de résoudre élégamment le problème du *bootstrap* en robotique évolutionniste en guidant le processus vers des étapes intermédiaires même si la fitness pour l'objectif principal est minimale. Néanmoins, cette approche nécessite la formulation d'objectifs intermédiaires évaluables en observant *uniquement* le comportement global de la solution, par exemple les mouvements d'un robot.

Le processus fonctionne différemment en ingénierie. Des briques de base, des modules, sont d'abord mises au point et constituent les étapes intermédiaires de la conception de l'artefact complexe. Chaque module est testé indépendamment en fonction de son cahier des charges puis intégré avec les autres modules. Dans la nature, cette démarche est proche de l'évolution par symbiose, comme cela a pu être le cas pour les eukaryotes (Margulis et Fester, 1991; López-Garcia et Moreira, 1999). Les explications actuelles de l'évolution de caractères complexes mettent cependant beaucoup plus en avant le rôle de l'exaptation (Gould et Vrba, 1982), c'est-à-dire de l'utilisation pour une fonction nouvelle d'un trait adapté initialement à une autre fonction. L'apparition des pattes avec des doigts à la place des nageoires chez les tétrapodes en constitue un exemple typique : bien qu'elles soient l'une des adaptations majeures qui ont facilité la transition de l'eau vers la terre, la morphologie des premiers

tétrapodes suggèrent qu'elles pourraient avoir été des adaptations à un environnement aquatique plutôt que terrestre (Coates et Clack, 1990). Ainsi, les pattes ont été soumises à une première pression sélective dans l'eau et il s'est ensuite avéré que ce « module » permettait de se mouvoir sur terre. La pression de sélection a alors changé pour adapter ces pattes à la locomotion terrestre. Si l'on se place du point de vue du problème du *bootstrap*, la locomotion terrestre n'aurait pas pu être envisagée si les pattes avec des doigts n'avaient pas été obtenue dans une étape précédente. La pression sélective intermédiaire a donc constitué la clef pour la conquête de la terre.

Dans la démarche modulaire en ingénierie, la symbiose ou l'exaptation, il existe toujours une pression sélective sur une des sous-parties qui précède la pression de sélection sur la capacité complexe. En biologie, ces pressions sont bien sûr identifiées a posteriori en tentant d'expliquer l'apparition de nouvelles espèces alors qu'en ingénierie elles sont définies lors de la phase de conception. Les sous-parties sont alors progressivement améliorées jusqu'à l'obtention des caractéristiques utiles pour accéder à l'étape suivante où une nouvelle pression sélective entre en jeu. Les bibliothèques logicielles continuent d'être améliorées même si elles sont utilisées dans beaucoup de projets, les pattes de certains amphibiens continuent à être adaptées à la nage, etc. La pression sélective permettant d'améliorer ces modules ne disparaît donc pas nécessairement.

Ce schéma de changements de pression de sélection lorsque c'est possible correspond à l'évolution incrémentale multiobjectif présentée au chapitre 3 où les sous-objectifs ne seraient évalués que pour certaines parties. Par exemple, un réseau de neurones contrôlant un pendule inversé a besoin de la dérivée du signal d'erreur. La pression de sélection sur ce calcul étant indirecte et les solutions basée sur un contrôle proportionnel simples et attractives, on peut envisager de diviser le réseau en modules et d'évaluer leur capacité à calculer la dérivée d'un signal. On peut ensuite définir un sous-objectif comme la performance du meilleur module. Ainsi, les individus possédant un module de dérivée efficace seront dominant quelle que soit la performance globale de la solution. Le processus évolutionniste pourra alors perfectionner ce module et essayer de construire des solutions au problème global l'exploitant. Le module de calcul de dérivée aura alors été *exapté* car d'abord développé sous une pression de sélection intermédiaire puis utilisée pour construire une solution performante pour un autre problème.

Dans ce chapitre, nous présentons une méthode originale d'évolution de réseaux de neurones qui s'inspire de ce principe et le combine à un génotype modulaire simple. La décomposition modulaire de Newman (2006a) est d'abord utilisée pour extraire des modules du réseau puis ceux-ci sont évalués en fonction des sous-objectifs. Ces modules peuvent être isolés dans le génotype, rendant la correspondance génotype-phénotype modulaire. Il est aussi possible de les échanger par l'opérateur de croisement afin que la descendance des meilleurs individus puisse combiner des modules efficaces. Enfin, on peut sub-

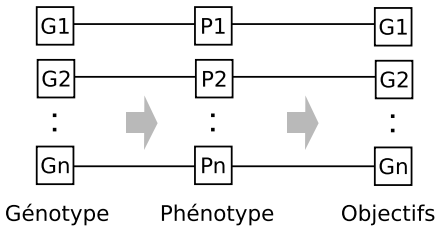



FIG. 5.1.: Alignement entre les modules du génotype, du phénotype et les gradients de sélection. À chaque module du génotype est associé un module du phénotype, à son tour relié à un objectif.

stituer un module à un autre au même nombre d'entrées et de sorties, ce qui permet d'exploiter le même module plusieurs fois dans le même individu.

Dans les travaux antérieurs sur l'évolution de réseaux de neurones modulaires (section 2.2 et particulièrement Hornby et al. (2003); Doncieux et Meyer (2004b); Gruau (1995)), les auteurs ont mis au point des représentations indirectes dans lesquelles des modules peuvent apparaître. Appliquant une pression de sélection sous la forme d'un unique objectif, ils supposent que le processus évolutionniste fera apparaître des modules en partant du principe que les solutions modulaires sont plus performantes. Cette hypothèse correspond à l'évolution pour l'évolvabilité (voir section 2.2) qui, d'après plusieurs expériences et travaux théoriques (voir Wagner et al. (2005)), ne semble pas être la force principale poussant à l'émergence de modules. Les hypothèses mettant en jeu plusieurs objectifs semblent une voie plus prometteuse pour obtenir des réseaux modulaires (Kashtan et Alon, 2005). L'approche que nous adoptons ici se distingue donc des approches antérieures par l'utilisation d'un lien explicite entre les modules, les fonctions qu'ils remplissent et les pressions de sélection. L'espace des modules est donc « aligné » (figure 5.1) avec les gradients de sélection, comme le suggèrent les expériences théoriques de Altenberg (2005) dont les conclusions sont décrites dans la section 2.5.2.

Ce chapitre est organisé en trois parties. Dans une première section, nous décrivons une expérience liminaire visant à comprendre si la modularité en tant que telle facilite l'évolution. Nous présentons ensuite l'approche modulaire incrémentale que nous proposons, en détaillant notamment le génotype utilisé. La dernière partie rapporte et analyse en profondeur l'application de cette approche pour faire évoluer des réseaux de neurones effectuant la fonction $\left[(a \oplus b) \wedge (c \oplus d)\right]$, abordée également dans le chapitre précédent.

## 5.2 EXPÉRIENCE LIMINAIRE

### 5.2.1 *Contexte expérimental*

Avant de s'intéresser aux liens entre modularité et sélection multiobjectif, on peut analyser l'influence de la modularité des réseaux de neurones sur les performances du processus évolutionniste. Puisque l'on dispose d'une mesure de la modularité, il est possible d'obtenir des réseaux modulaires en ajoutant simplement un objectif correspondant à la modularité du réseau. Ainsi, entre deux réseaux aux performances égales, le processus de sélection choisira la solution la plus modulaire. Si le simple fait d'être modulaire affecte l'évolvabilité, l'ajout de l'objectif de modularité devrait augmenter significativement les performances de l'évolution.

Pour tester cette hypothèse, nous avons utilisé la fonction booléenne $\left[(a \oplus b) \wedge (c \oplus d)\right]$, déjà employée dans la section 4.3.1. Cette fonction est intéressante car elle est suffisamment complexe pour ne pas être trouvée en quelques générations mais constituée de deux modules xor, qui sont des fonctions logiques simples (figure 4.1). Nous sommes donc sûr qu'il existe une solution modulaire.

La fitness d'un individu $x$ est définie comme la somme des erreurs pour chaque jeu d'entrée à laquelle on adjoint le score de modularité $Q$ calculé avec la méthode de Newman (2006a) :

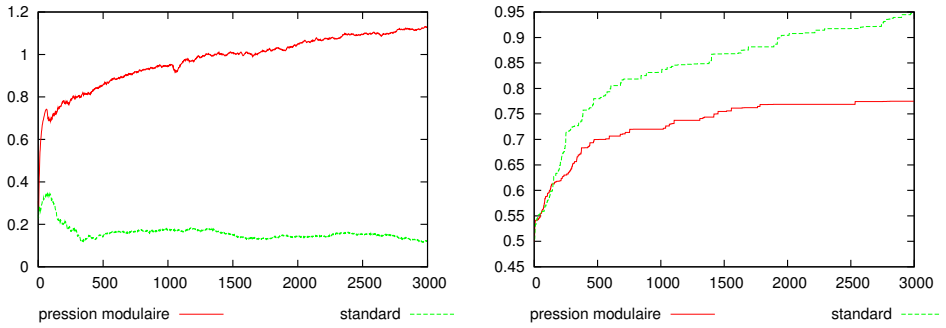
$$\begin{cases} F_x = 1 - \frac{1}{16} \sum_{i=1}^{16} |o_i - d_i| \\ Q_x \end{cases}$$

où $o_i$ est la sortie du réseau de neurones pour le jeu d'entrée $i$ et $d_i$ la sortie attendue.

Un codage direct sans contraintes sur la topologie dont les paramètres sont décrits dans l'annexe A.2.3 a été utilisé avec l'algorithme NSGA-II et une population de 400 individus. Le croisement n'est pas utilisé. 20 expériences ont été lancées.

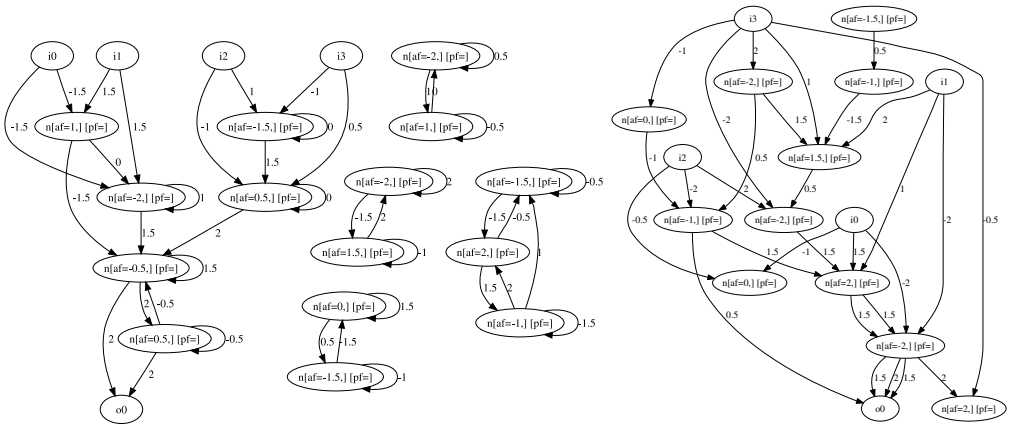
### 5.2.2 *Résultats*

Lorsque l'on n'utilise pas d'objectif de modularité, la figure 5.2(a) montre que la modularité moyenne de la population baisse puis se stabilise vers 0.2 (une valeur faible). À l'inverse, lorsqu'on l'utilise, il augmente progressivement et se stabilise aux alentours de 1.2 (une valeur forte). La modularité est donc bien optimisée par le processus évolutionniste. Cependant, la figure 5.2(b) montre que l'augmentation de modularité ne se traduit pas par une hausse de la fitness. Au contraire, les solutions non modulaires obtiennent plus rapidement une fitness élévée et convergent dans beaucoup plus de situations. Favoriser les solutions modulaires ne permet donc pas d'augmenter les performances de l'évolution, c'est-à-dire l'évolvabilité, mais, au contraire,




(a) (b)

FIG. 5.2.: (a) Modularité moyenne de la population en fonction de la génération pour le problème $\left[(a \oplus b) \wedge (c \oplus d)\right]$, avec et sans objectif de modularité (moyenne sur 20 expérience). (b) Fitness moyenne en fonction de la génération (moyenne sur 20 expériences).




(a) (b)

FIG. 5.3.: (a) Exemple de réseau obtenant une fitness maximum (1) lorsque l'objectif de modularité est utilisé. (b) Exemple de réseau obtenant une fitness maximum sans objectif de modularité.

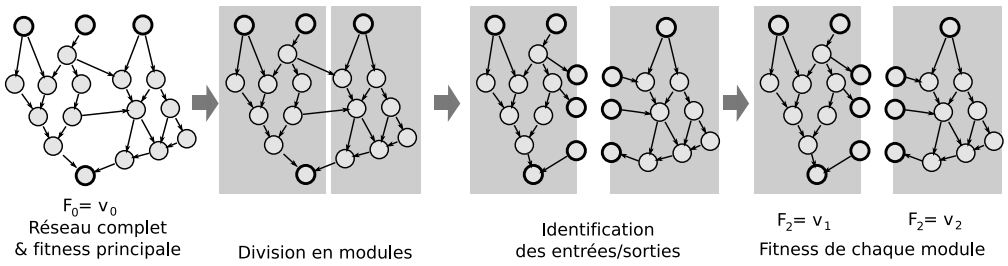



FIG. 5.4.: Principe de l'alignement entre modules et objectifs. Une fois les modules et leurs entrées/sorties identifiés, la performance de chaque module pour chacun des modules peut être calculée « in-vivo » en observant les valeurs de sorties des neurones de sortie du module pendant l'évaluation de l'objectif principal ou « in-vitro » en le soumettant à des jeux de test particuliers.

la diminue. Cette conclusion va dans le sens d'expériences précédentes où les solutions modulaires sont plus dures à obtenir que les autres (Bowers et Bullinaria, 2005).

On peut cependant constater que lorsque l'évolution converge avec l'objectif de modularité, elle mène à des solutions modulaires correspondant à l'intuition (figure 5.3). Ainsi, sur la figure 5.3(a), on distingue clairement deux modules effectuant chacun la fonction « xor ». Les entrées 0 et 1 ($i0$ et $i1$) sont regroupées dans un module distinct de celui associant $i1$ et $i2$, puis ces modules sont reliés à la sortie par quelques neurones effectuant une fonction « et ». Afin d'augmenter la modularité des réseaux, l'évolution a ajouté quatre sous-réseaux, déconnectés des entrées et des sorties et donc sans influence sur la fitness principale. Ces sous-réseaux ne sont pas connectés entre eux alors que chacun de leurs neurones sont très connectés, ils constituent donc des modules évidents. Le nombre de neurones n'étant pas fixé, l'évolution peut ainsi artificiellement augmenter la modularité à l'infini. Les réseaux obtenus sans objectif de modularité sont beaucoup moins structurés (figure 5.3(b)) et, notamment, les entrées ne semblent pas regroupées logiquement comme c'est le cas sur la figure 5.3(a).

Ces résultats montrent que s'il est possible d'obtenir des réseaux à la structure similaire à celle de la fonction objectif en maximisant la modularité, cette maximisation en tant que telle ne constitue pas une bonne heuristique pour améliorer les performances des algorithmes évolutionnistes.

## 5.3 ÉVOLUTION INCRÉMENTALE MODULAIRE

### 5.3.1 *Principe*

Comme nous l'avons vu dans les sections 2.2 et 2.5.2, pour bénéficier des avantages de la modularité pour le processus évolutionniste, il est nécessaire d'aligner génotype, phénotype et sélection (Altenberg, 2005). L'alignement génotype-phénotype doit prendre la forme d'une correspondance modulaire, c'est-àdire que la modification d'un module du génotype ne doit influer que sur le module phénotypique associé. D'après Wagner (1996), seulement deux types de phénomènes peuvent augmenter la modularité de la correspondance génotype-phénotype et donc de leur alignement (figure 2.5) :

1. la parcellation, qui consiste en l'élimination des effets pléiotropiques entre les membres de différents modules et l'augmentation ou la maintenance des effets pléiotropiques intra-modules ;
2. l'intégration, qui consiste en la création d'effets pléiotropiques parmi des caractères initialement indépendants mais partageant une fonction similaire ; à l'extrême, l'intégration correspond à remplacer un module du phénotype par l'expression d'un module du génotype déjà utilisé.

Bien que l'existence de la parcellation et de l'intégration semble largement confirmée, les mécanismes moléculaires sur lesquels ils s'appuient sont complexes et encore mal connus. De manière très abstraite, on peut considérer que la parcellation revient à découper le génome en parties n'intéragissant plus et l'intégration à la réutilisation d'un même ensemble de gènes lorsque c'est possible. Ces deux phénomènes peuvent être contrebalancés par la différentiation des éléments répétés (Muller et Wagner, 1991) qui revient à modifier le génotype de manière à générer des variantes des phénotypes intégrés.

Il est possible d'appliquer ces trois concepts abstraits à l'évolution de réseau de neurones, permettant ainsi d'obtenir une correspondance génotype-phénotype modulaire. Supposons que l'on dispose d'un réseau de neurones décrit via un codage direct. Une mutation, par exemple l'ajout d'un arc, peut changer la division modulaire du réseau de neurones, les modules phénotypiques sont donc très instables et les effets pléiotropiques importants. La parcellation correspond à extraire un module du réseau, et à rendre son génotype indépendant. Elle fixe donc le découpage et rend le module stable dans le temps. Ainsi, on limite les effets pléiotropiques des mutations, celles intervenant sur le module parcellé n'affectant pas le phénotype du reste du réseau et réciproquement. L'intégration peut être vue comme le remplacement d'un ou plusieurs modules du réseau principal ayant le même nombre d'entrées et sorties qu'un module résultat d'une parcellation par une référence vers ce dernier module. Ainsi, toute mutation affectant le module parcellé affectera aussi le ou les modules intégrés. Enfin, la différentiation peut correspondre à réinsérer un module parcellé ou intégré dans le réseau principal, lui donnant ainsi la possibilité d'évoluer indépendamment. Si cette différentiation ressemble vraisem-

blablement peu à son homologue biologique, elle a l'avantage de permettre d'annuler les effets de la parcellation et de l'integration. Cette interprétation des principes de parcellation, intégration et différentiation aux réseaux de neurones permet de définir un codage indirect simple que nous décrirons en détail dans la section suivante.

Ces principes sont présents à différents degrés et sous une forme moins explicite dans les codages modulaires proposés antérieurement. ModNet (Doncieux et Meyer, 2004a), par exemple, utilise une liste de modules-modèles, une liste de modules effectivement utilisés et une liste de liens entre les modules. Par conséquent, les changements affectant un module n'ont pas d'influence sur les autres et le génome est donc proche de celui que nous décrivons ici après une parcellation. Les modules-modèles pouvant être utilisés plusieurs fois, un phénomène d'intégration est également présent. La différentiation, par contre, n'a pas d'équivalent. En outre, les modules de ModNet sont restreints à une entrée et une sortie et n'ont pas de liens explicite avec des modules que nous pourrions déduire de l'observation du phénotype. ModularNEAT (Reisinger et Miikkulainen, 2007), en faisant évoluer symbiotiquement une population de modules et une de plans de construction, impose également le découpage modulaire au niveau du génotype, se passant ainsi de la parcellation pour établir une correspondance génotype-phénotype modulaire. La répétition d'un module peut être vue comme une intégration mais ModularNEAT ne propose pas d'équivalent à la différentiation. Dans les deux cas, la différence principale entre le génotype que nous proposons et ces deux méthodes réside dans le peu de liens entre module génotypique et module phénotypique car la correspondance est fixée par le génotype, sans possible interaction avec les modules du phénotype.

Si aucun argument théorique ne permet de rejeter ces codages, deux raisons nous ont poussé à en utiliser un autre. En premier lieu, nous souhaitons nous appuyer sur un codage le plus simple possible afin de nous concentrer sur les liens entre modularité et pression de sélection. La complexité amenée par les multiplies populations et la méthode NEAT ainsi que les limitations de ModNet constituent des obstacles pour analyser les phénomènes qui nous intéressent. En second lieu, nous espérons effectuer un parallèle explicite avec la littérature en biologie concernant l'évolution de la modularité afin de pouvoir l'utiliser au mieux.

L'alignement des gradients de sélection sur les modules du phénotype peut être effectué en s'inspirant du chapitre 3. Un ou plusieurs sous-objectifs sont ajoutés à l'objectif principal, chacun correspondant au gradient permettant de sélectionner une « brique intermédiaire » susceptible d'être utile à la réalisation de la fonction principale. Afin d'évaluer ces objectifs sur chacun des sous-modules, il faut commencer par découper le réseau de neurones en modules. Les objectifs peuvent ensuite être évalués soit « in-vivo » en observant la sortie des modules pendant l'évaluation de l'objectif principal, soit « in-vitro » en isolant le module du reste du réseau et en le soumettant à des jeux d'en-

trée spécifiques (figure 5.4). À notre connaissance, aucune méthode antérieure pour l'évolution de réseau de neurones n'a tenté de créer un tel lien entre sélection et modules. Les travaux les plus proches sont peut-être ceux concernant l'évolution symbiotique (section 2.3.5) dans lesquels la fitness des modules est attribuée à partir de leur contribution à la fitness globale. Dans ces méthodes, il existe bien une pression de sélection sur les modules mais elle reste indirecte.

### 5.3.2 *Codage et opérateurs génétiques*

Afin de pouvoir implémenter simplement et explictement les principes de parcellation, intégration et différentiation, nous avons défini le génotype d'un réseau de neurones comme l'agrégation d'un réseau principal et d'une liste de sous-réseaux (figure 5.5). Chaque sous-réseau est associé à une ou plusieurs listes de liens permettant de relier chacune de ses entrées/sorties à un neurone du réseau principal. La présence de plus d'une liste de liens signifie que le sous-réseau doit être instancié plusieurs fois, à chaque fois avec des liens différents. Les réseaux sont directement représentés par un graphe orienté avec des arcs pondérés et dont certains sommets sont arbitrairement choisis comme entrée/sortie.

Lors de la première génération, des graphes comportant un nombre spécifié de neurones d'entrées et de sortie sont générés aléatoirement, sans sous-réseaux. Les individus des premières générations sont donc constitués uniquement du réseau principal. Pendant l'évolution, le réseau principal est d'abord décomposé en modules via la méthode de Newman (2006a) puis leurs entrées/sorties sont déterminées. On considère que l'entrée d'un module est constituée soit par un arc entrant reliant un neurone extra-module à un neurone intra-module, soit par une entrée du réseau principal. De manière similaire, une sortie est soit un arc sortant reliant un neurone intra-module à un neurone extra-module, soit par une sortie du réseau principal. Une fois les modules identifiés, cinq opérateurs de mutation sont utilisés :

- *mutation structurelle* des graphes du génotype :
  - ajout / suppression d'une connexion ;
  - ajout / suppression d'un sommet ;
- *mutation paramétrique* : changement de la pondération d'un ou plusieurs arcs dans les réseau principal et les sous-réseaux ;
- *parcellation* (figure 5.6) : un module est sorti du réseau principal ; il devient un sous-réseau et la liste de ses entrées et sorties lui est associée dans le génotype ; pour augmenter l'efficacité de cet opérateur, il est possible de choisir en priorité des modules ayant de bonnes fitness pour les sous-objectifs ;
- *intégration* (figure 5.7) : un module ayant le même nombre d'entrées/sorties qu'un des sous-réseaux du génotype est éliminé du réseau principal et la liste de ses entrées/sorties est ajoutée à la liste de liens du sous-réseau du génotype ;

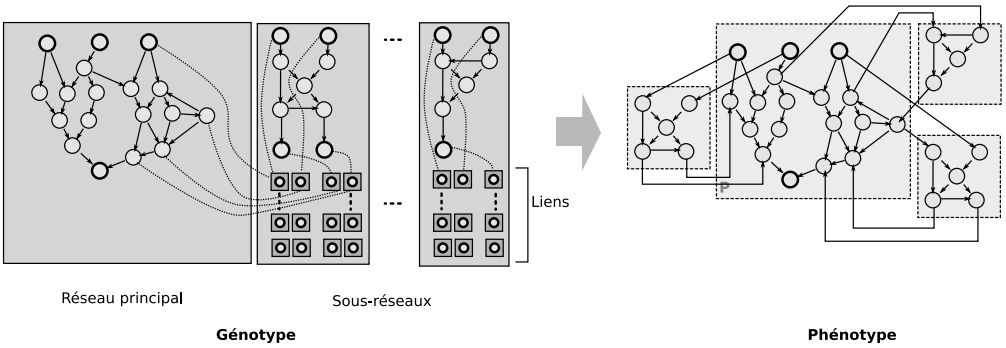



FIG. 5.5.: Le génotype est constitué d'un réseau principal et d'un ou plusieurs sous-réseaux. Chaque sous-réseau est associé à une liste de liens permettant de le relier au réseau principal. Si plusieurs liste de liens existent, le réseau doit être ajouté plusieurs fois au réseau principal. Le phénotype présenté est un exemple du type de réseau obtenu en développant le génotype.

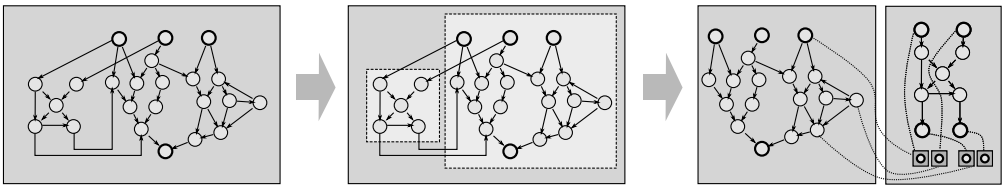


FIG. 5.6.: Opérateur de parcellation. Un module est supprimé du réseau principal pour devenir un nouveau sous-réseau. Les entrées et sorties du module sont ajoutées dans la liste des liens du sous-réseau.

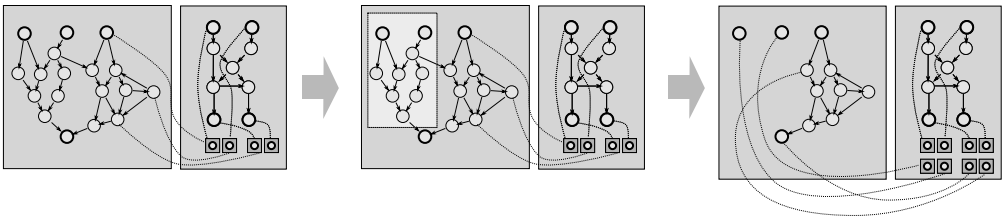


FIG. 5.7.: Opérateur d'intégration. Un module au même nombre d'entrées/sorties qu'un des sous-réseaux est supprimé du réseau principal et la liste de ses entrées/sorties est ajoutée à la listes des liens du sous-réseau.

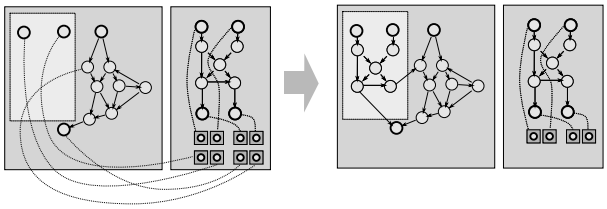


FIG. 5.8.: Opérateur de différentiation. Un module présent parmi les sous-réseaux est réintégré au réseau principal en utilisant la liste de liens correspondante, qui est ensuite retirée des listes associées au sous-réseau. Si aucune liste de liens n'existe pour un sous-réseau, il est supprimé du génotype.

– *différentiation* (figure 5.8) : un sous-réseau de la liste des sous-réseaux est ajouté au réseau principal en utilisant la liste de liens correspondante, qui est ensuite retirée des listes associées à ce sous-réseau ; si aucune liste de liens n'existe pour un sous-réseau, il est supprimé du génotype.

La différentiation et l'intégration dépendent de la parcellation, ce qui impose à l'évolution un ordre d'application des opérateurs. Un module est d'abord isolé par la parcellation, puis l'évolution peut essayer de l'utiliser plusieurs fois, via l'intégration, ou de le dissoudre dans le réseau, si la parcellation se révèle mener l'évolution vers une impasse.

Il est possible d'implémenter un croisement rudimentaire en échangeant simplement deux modules isolés par parcellation aux mêmes nombres d'entrées/sorties. Ainsi, un module efficace peut être propagé rapidement dans la population.

L'approche présentée dans ce chapitre est compatible avec le maintien de la diversité comportementale du chapitre 4. Nous essaierons donc dans la suite de déterminer s'il améliore les résultats comme c'était le cas dans le chapitre 4.

## 5.4 EXPÉRIENCES ET RÉSULTATS

### 5.4.1 *Contexte expérimental*

Nous avons testé l'approche proposée dans ce chapitre sur la fonction $\left[(a \oplus b) \wedge (c \oplus d)\right]$, comme dans les sections 4.3.1 et 5.2. Cette fonction est rapide à évaluer, possède une structure modulaire évidente et utilise deux sous-modules identiques. Les opérateurs que nous venons de définir devraient donc pouvoir mener le processus évolutionniste vers des réseaux de neurones calculant correctement cette fonction.

La même fitness $F_x$ que dans les sections précédentes est utilisée (la somme des erreurs) mais un deuxième objectif $F_m$ est ajouté pour appliquer une pression de sélection sur les sous modules calculant la fonction xor. De plus, un troi-

sième objectif correspondant à la diversité comportementale (voir le chapitre 4) peut aussi être utilisé. Nous avons donc le problème multiobjectif suivant :

$$\begin{cases} F_x = 1 - \frac{1}{16}\sum_{i=1}^{16} |o_i - d_i| \\ F_m = \max_{m \in M} F_{xor}(m) \\ F_{div} \end{cases}$$

où $o_i$ est la sortie du réseau de neurones après développement pour le jeu d'entrée $i$, $d_i$ la sortie attendue et $M$ l'ensemble des modules du réseau principal et des sous-réseaux.

$F_{xor}(m)$ correspond à la somme des erreurs pour les 4 jeux d'entrées possibles pour la fonction *xor* et le réseau de neurone correspondant au module $m$, où l'on considère comme « compatible » les modules $m$ possédant 2 entrées et 1 sortie, :

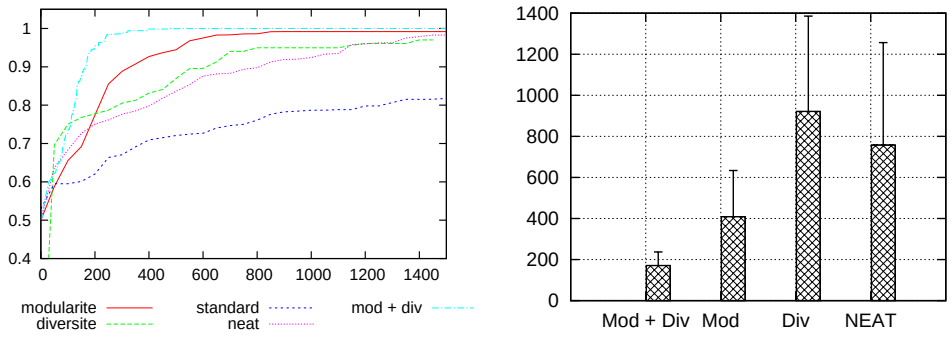
$$F_{xor}(m) = \begin{cases} 1 - \frac{1}{4}\sum_{i=1}^{4} |o_i - d_i| \text{ si m est compatible} \\ -1 \text{ sinon} \end{cases}$$

Nous nous attendons à ce qu'un module possédant deux entrées et une sortie soit d'abord isolé par parcellation, celle-ci ayant lieu en priorité pour des modules performants pour $F_{xor}$ et donc avec une fitness différente de $-1$. La pression de sélection sur ce module devrait le mener rapidement à effectuer correctement la fonction xor. Il pourra alors être dupliqué par intégration. La réunion des deux modules par quelques neurones effectuant un « et » logique pourrait ainsi permettre l'obtention de réseaux de neurones à la structure proche de celle sous-jacente de la fonction objectif.

L'algorithme NSGA-II et une population de 400 individus ont été utilisés avec les paramètres décrits dans la section A.2.3. Lors de la parcellation, le module choisi est forcément celui qui a le meilleur score pour $F_m$. Deux variantes ont été testées, avec et sans objectif de diversité comportementale. Pour chacune d'elles, 30 expériences ont été lancées.

### 5.4.2 *Résultats*

La figure 5.9(a) compare l'évolution de la moyenne des meilleures fitness pour les deux variantes avec une expérience témoin utilisant uniquement un codage direct, NEAT et le maintien de la diversité décrit dans le chapitre 4. Une fitness supérieure à 0.95, c'est-à-dire correspondant à des individus ayant tous convergés vers une solution avec la bonne réponse pour tous les jeux d'entrées, est atteinte en moins de 200 générations avec l'objectif de diversité et environ 400 lorsqu'il n'est pas utilisé. L'utilisation d'un codage modulaire combinée à la sélection multiobjectif apparaît donc bien plus efficace que l'évolution non-modulaire « standard » mais se révèle aussi plus rapide à converger que NEAT et l'approche basée sur la diversité de comportement du chapitre 4. Alors que moins de 200 générations sont requises dans la meilleure des variantes, l'approche basée sur la diversité en a besoin de 900 et NEAT de 750 (figure 5.9(b)).




(a) (b)

FIG. 5.9.: (a) Évolution de la moyenne des meilleures fitness pour les 30 expériences. (b) Génération moyenne de convergence (fitness supérieure à 0.95). Les différences sont toutes statistiquement significatives sauf entre la diversité comportementale et NEAT (annexe A.1.1).

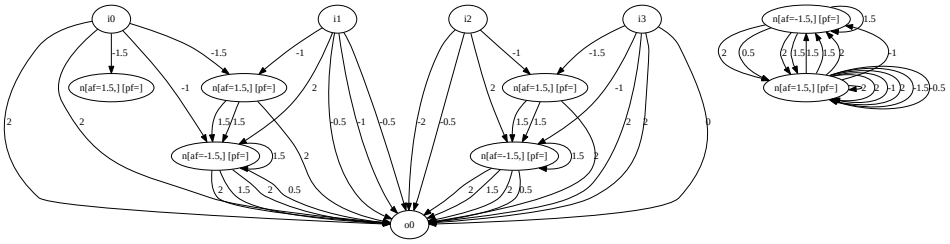



FIG. 5.10.: Réseau de neurones obtenu réalisant la fonction booléenne $\big[(a \oplus b) \wedge (c \oplus d)\big]$.

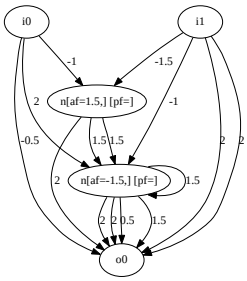



FIG. 5.11.: Module du réseau de la figure 5.10 isolé par parcellation. Il calcule un ou exclusif (xor) entre ses deux entrées.

La figure 5.10 montre un exemple de réseau avec une fitness de 1.0 où l'on distingue clairement la répétition d'un module effectuant la fonction xor (figure 5.11). Le génotype correspondant utilise un sous-réseau instancié deux fois.

### 5.4.3 *Influence de chaque composant*

Nous avons analysé l'effet des quatre opérateurs génétiques – parcellation, intégration, différentiation et croisement – et de l'ajout de $F_m$ en observant les taux de convergence et les fitness moyennes obtenues dans chacune des $2^5 = 32$ combinaisons possibles. L'objectif de diversité n'a pas été utilisé afin de concentrer l'analyse sur l'approche modulaire. Un telle méthode s'inspire des expériences de lésions simples présentes dans certains travaux sur l'évolution de réseaux de neurones (voir par exemple Stanley et Miikkulainen (2002a)) où les variantes analysées avec et sans chaque opérateur sont comparées. Dans ce dernier cas, seul l'effet indépendant des opérateurs peut être compris ; en explorant toutes les combinaisons, on peut analyser l'effet des associations d'opérateurs. En remarquant que les opérateurs de croisement, différentiation et intégration sont inopérants si la parcellation n'a pas été activée, 18 expériences doivent être lancées. Nous les avons chacune effectuées 30 fois pour 1500 générations avec les paramètres de la section précédente, puis nous avons calculé le taux de convergence moyen et la fitness moyenne. Les résultats sont consignés dans la table 5.1.

L'utilisation d'aucun opérateur seul semble être clairement corrélée avec la performance, suggérant que les bons résultats obtenus dans certaines expériences nécessitent l'utilisation simultanée de plusieurs opérateurs. Le multiobjectif seul, c'est-à-dire l'alignement des modules du phénotype avec la sélection, semble même être néfaste pour le taux de convergence en dépit du « guide » supplémentaire fourni par le deuxième objectif (3% contre 44% pour l'expérience témoin, c'est-à-dire celle sans multiobjectif et sans opérateurs autres que ceux du codage direct). Cette contre-performance peut s'expliquer par l'utilisation d'une partie de la population pour couvrir l'ensemble des compromis. Néanmoins, l'analyse de l'évolution de la modularité dans le cas simple et multiobjectif, sans autres opérateurs, montre que l'utilisation de l'objectif $F_m$ augmente la modularité (en moyenne 0.38 après 1500 générations contre 0.18) sans pour autant permettre l'émergence rapide d'un module xor fonctionnel (en moyenne, une fitness de seulement 0.64 a été obtenue pour les modules xor). Cette observation rejoint celles de la section 5.2 qui montraient que l'augmentation de la modularité ne s'accompagne pas nécessairement d'une augmentation du taux de convergence. L'ajout de la parcellation permet d'obtenir des modules xor fonctionnels en environ 300 générations, ce qui représente une amélioration très significative et souligne l'intérêt de rendre les modules stables. Cependant, l'obtention rapide de ces modules ne suffit pas à mener à des réseaux efficaces, seules 19% des expériences convergeant contre 44% dans l'expérience témoin. Les quatre opérateurs génétiques semblent être

| M | P | C | I | D | taux convergence | Fitness moyenne |
|---|---|---|---|---|---|---|
| | | | | | 0.438 | 0.876 (0.116) |
| | | | | • | 0.438 | 0.876 (0.116) |
| | | | • | | 0.438 | 0.876 (0.116) |
| | | | • | • | 0.438 | 0.876 (0.116) |
| | | • | | | 0.438 | 0.876 (0.116) |
| | | • | | • | 0.438 | 0.876 (0.116) |
| | | • | • | | 0.438 | 0.876 (0.116) |
| | | • | • | • | 0.438 | 0.876 (0.116) |
| | • | | | | 0.25 | 0.835 (0.117) |
| | • | | | • | 0.312 | 0.842 (0.12) |
| | • | | • | | 0.375 | 0.861 (0.112) |
| | • | | • | • | 0.281 | 0.858 (0.103) |
| | • | • | | | 0.344 | 0.863 (0.114) |
| | • | • | | • | 0.312 | 0.858 (0.107) |
| | • | • | • | | 0.438 | 0.877 (0.116) |
| | • | • | • | • | 0.344 | 0.86 (0.108) |
| • | | | | | 0.031 | 0.739 (0.066) |
| • | | | | • | 0.031 | 0.739 (0.066) |
| • | | | • | | 0.031 | 0.739 (0.066) |
| • | | | • | • | 0.031 | 0.739 (0.066) |
| • | | • | | | 0.031 | 0.739 (0.066) |
| • | | • | | • | 0.031 | 0.739 (0.066) |
| • | | • | • | | 0.031 | 0.739 (0.066) |
| • | | • | • | • | 0.031 | 0.739 (0.066) |
| • | • | | | | 0.188 | 0.797 (0.116) |
| • | • | | | • | 0.406 | 0.851 (0.125) |
| • | • | | • | | 1.0 | 1.0 (0.0) |
| • | • | | • | • | 0.969 | 0.992 (0.043) |
| • | • | • | | | 0.719 | 0.936 (0.102) |
| • | • | • | | • | 0.656 | 0.927 (0.103) |
| • | • | • | • | | 0.938 | 0.988 (0.048) |
| • | • | • | • | • | 0.969 | 0.988 (0.065) |

TAB. 5.1.: Influence de chaque composant sur le taux de convergence moyen et la fitness moyenne. 30 expériences de chaque type ont été lancées sauf pour les combinaisons correspondant aux lignes sur fond gris, qui ont été déduites à partir des autres résultats en considérant que la parcellation était un prérequis pour certains opérateurs.

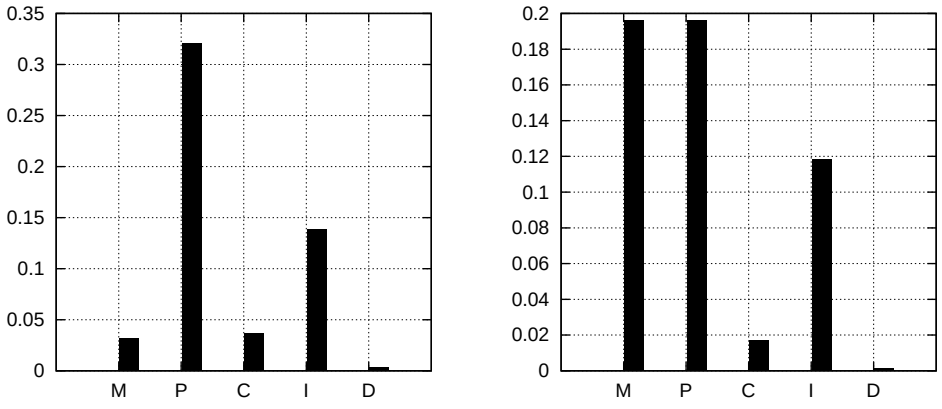



FIG. 5.12.: (a) Valeurs de Shapley calculées en utilisant le taux de convergence brut $V_c(S)$ de chaque combinaison. (b) Valeurs de Shapley calculées en utilisant le taux modifié $V(S)$.

néfastes aux performances s'ils ne sont pas combinés avec le multiobjectif car toutes les expériences correspondantes obtiennent des taux de convergences inférieurs à celui du témoin. Le maintien des modules, leur échange et leur répétition, c'est-à-dire l'utilisation d'un codage modulaire, ne semble donc pas être suffisants pour améliorer les performances du processus évolutionniste. Au final, les 4 expériences avec plus de 90% de taux de convergence ont pour point commun la combinaison du multiobjectif, de la parcellation et de l'intégration.

La théorie des jeux permet d'analyser plus précisément l'influence de chaque opérateur. En considérant chacun des constituants présentés comme des joueurs d'un jeu coalitionnel, estimer leur contribution sur les performances du processus évolutionniste revient à répartir équitablement les gains, c'est-à-dire la fitness finale ou le taux de convergence, entre eux. En théorie des jeux et en économie, la valeur des gains équitablement assignée à chaque joueur correspond à la *valeur de Shapley* (Shapley, 1953), définie à partir de l'*importance marginale* d'un joueur $i$ pour une coalition $S$, avec $i \notin S$ et une fonction de valeur $v(S)$ :

$$\Delta_i(S) = v(S \cup \{i\}) - v(S)$$

Si on désigne par $R$ l'ensemble des $n!$ arrangements des joueurs, par $S_i(r)$ l'ensemble des joueurs précédant $i$ dans $R$ et $N$ l'ensemble des joueurs, la valeur de Shapley s'écrit :

$$\gamma_i(N,v) = \frac{1}{n!} \sum_{r \in R} \Delta_i(S_i(r))$$

Cette équation peut être reformulée pour être plus simple à calculer, on obtient alors (voir Keinan et al. (2004)) :

$$\gamma_i(N,v) = \underbrace{\frac{1}{n!}\sum_{S\subset N, i\in S} v(S)\cdot(|S|-1)!\cdot(n-|S|)!}_{\text{configurations avec i}} - \underbrace{\frac{1}{n!}\sum_{S\subset N, i\notin S} v(S)\cdot(|S|)!\cdot(n-|S|-1)!}_{\text{configurations sans i}}$$

Plusieurs choix sont possibles pour $v(S)$, c'est-à-dire pour évaluer la valeur d'une configuration particulière du processus évolutionniste. Le plus simple est d'utiliser le taux de convergence obtenu avec cette configuration (figure 5.12(a)). Si on retrouve l'importance de la parcellation et celle de l'intégration, la faible valeur de l'utilisation du multiobjectif est surprenante. L'analyse du tableau 5.1 permet de comprendre que si le multiobjectif apparaît indispensable à l'obtention de très bonnes valeurs, sa contribution moyenne parmi toutes les configurations est très faible, voire négative. Il obtient donc une valeur de Shapley très faible en comparaison de la parcellation qui participe à des configurations avec de bons résultats et se révèle beaucoup moins néfaste que le multiobjectif dans la plupart des cas.

Si les résultats obtenus avec cette fonction de valeur semblent en contradiction avec le tableau 5.1, c'est simplement parce que la fonction de valeur choisie ne correspond pas avec ce que l'on regarde dans le tableau. En effet, nous sommes en priorité intéressés par expliquer les expériences qui améliorent le taux de convergence par rapport à l'expérience témoin. Le fait que certaines configurations de composants utilisés dans le processus évolutionniste puisse réduire ses performances ne nous intéresse pas. Pour modéliser ce concept et mieux utiliser la valeur de Shapley, on peut simplement transformer la fonction de valeur pour considérer comme égales toutes les expériences n'améliorant pas le taux de convergence par rapport à l'expérience témoin. Ainsi, on rend asymétrique la prise en compte des contributions de chaque composant en ne s'intéressant pas aux contributions négatives. Nous utilisons donc la fonction de valeur $v(S)$ à la place du taux de convergence $v_c(S)$ :

$$v(S) = \max(v_c(\emptyset), v_c(S))$$

Les valeurs de Shapley obtenues avec cette fonction de valeur sont tracées sous forme d'histogramme sur la figure 5.12(b). Les deux plus grandes valeurs sont obtenues par le multiobjectif et la parcellation, montrant qu'ils sont les deux composants les plus importants du système évolutionniste que nous analysons. En outre, les deux valeurs semblent identiques, ce qui suggère que les deux sont nécessaires mais que l'un ou l'autre isolé n'apporte rien. Cette hypothèse peut être facilement confirmée par l'analyse du tableau 5.1 : les

seules configurations menant à un taux de convergence supérieurs à l'expérience témoin sont ceux utilisant à la fois le multiobjectif et la parcellation[1]. Le deuxième opérateur avec une valeur de Shapley importante est l'intégration, dont l'intérêt est évident pour la fonction logique testé. Sa valeur reste moins importante que les deux précédents opérateurs, ce qu'il est difficile d'établir à la seule lecture du tableau 5.1. Cet opérateur pourrait bien sûr être moins utile pour d'autres tâches où la répétition d'un module s'avérerait peu avantageuse. Le croisement a une valeur de Shapley beaucoup plus faible mais positive, montrant que son influence est faible mais qu'il ne baisse pas les performances. La valeur quasi-nulle de la différentiation montre qu'elle n'a rien apporté au processus dans cette expérience. Elle pourrait cependant se révéler utile dans d'autres tâches si l'évolution est bloquée.

En conclusion, toutes les expériences n'utilisant pas simultanément la parcellation, c'est-à-dire l'alignement génotype-phénotype et le multiobjectif, c'est-à-dire l'alignement phénotype-sélection, obtiennent au mieux des performances équivalentes à l'expérience témoin. L'analyse des opérateurs démontre donc clairement que, au moins pour la tâche envisagée ici, il est *nécessaire* d'aligner modules du génotype, modules du phénotype et sélection.

## 5.5 CONCLUSION

L'évolution des êtres vivants, via l'exaptation et la symbiose, et l'ingénierie, via la conception modulaire, s'appuient sur la mise au point de modules avant de les intégrer dans des systèmes plus complexes. Dans les deux démarches, il existe une pression de sélection plus ou moins explicite sur les modules qui forme ainsi une marche sur laquelle peut s'appuyer le processus de conception. Cette vision de l'évolution est proche de l'évolution incrémentale que nous avons abordée dans le chapitre 3 mais cette dernière ne permet d'exercer des pressions sélectives que sur les comportements intermédiaires et non pas les sous parties potentiellement utiles. Dans ce chapitre, nous avons donc proposé une méthode pour permettre une évolution incrémentale *et* modulaire.

Des études théoriques (Altenberg, 2005) montrent la nécessité d'aligner les modules du génotype, du phénotype et de la sélection. Nous avons essayé de transposer ce principe d'évolution modulaire à l'évolution de réseaux de neurones. Pour cela, nous nous sommes d'abord appuyés sur les concepts de

1 On peut aisément montrer l'égalité des deux valeurs en détaillant le calcul. Afin de simplifier les calculs sans perte de généralité, on peut poser $V_c(S) = v_c(S) - v_c(\emptyset)$. En s'appuyant sur les données du tableau 5.1, beaucoup de termes deviennent nuls. En définissant chaque configuration par un nombre binaire, on a :

$$\begin{aligned}\gamma_M &= \frac{1}{5!}\left(4V_c(11010) + 12V_c(11011) + 12V_c(11101) + 6V_c(11110) + 24V_c(11111)\right)\\ \gamma_P &= \frac{1}{5!}\left(4V_c(11010) + 12V_c(11011) + 12V_c(11101) + 6V_c(11110) + 24V_c(11111)\right)\end{aligned}$$

parcellation, d'intégration et de différentiation (Wagner, 1996) pour mettre au point un génotype simple permettant d'établir une correspondance génotype-phénotype modulaire. Nous proposons ensuite d'établir la correspondance entre modules phénotypiques et gradients de sélection en associant des sous-objectifs à des modules issus d'une décomposition modulaire du réseau (Newman, 2006a) ou identifiés comme tels dans le génotype. Ainsi, génotype, sélection et phénotype sont alignés.

Nous avons testé cette approche en faisant évoluer des réseaux de neurones capables d'approcher la fonction booléenne $(A \text{ xor } B) \text{et} (C \text{ xor } D)$. Un objectif ajoutant une pression sélective sur les modules effectuant la fonction *xor* a été utilisé et un autre pour le maintien de la diversité comportementale (chapitre 4). Les résultats obtenus montrent que :

- la méthode modulaire obtient des résultats significativement meilleurs que les autres méthodes testées (témoin, NEAT, diversité uniquement) en nécessitant environ deux fois moins de générations pour converger ;
- la méthode modulaire se combine efficacement avec le maintien de la diversité, divisant encore par deux le nombre de générations requises pour converger ;
- l'analyse des opérateurs à l'aide de la valeur de Shapley montre que la parcellation et le multiobjectif sont tous deux nécessaires pour obtenir de bonnes performances alors qu'isolés ils n'amènent aucun gain ;
- ce dernier point montre que, au moins dans la tâche analysée, il est nécessaire d'aligner phénotype, génotype et sélection pour profiter des avantages de la modularité.

Ces résultats contredisent les hypothèses sur lesquelles s'appuient les travaux précédents (Hornby, 2005; Reisinger et al., 2004; Doncieux et Meyer, 2004a; Gruau, 1995) sur l'évolution de réseaux de neurones modulaires, ceux-ci supposant que la modularité de la correspondance génotype-phénotype suffit à augmenter les performances du processus évolutionniste.

# 6

# DISCUSSION ET PERSPECTIVES

RÉSUMÉ. Après avoir synthétisé les contributions de cette thèse, nous présentons des exemples d'applications potentielles des méthodes proposées en robotique évolutionniste. Nous analysons ensuite les conséquences de l'ajout de nombreux objectifs dans une optimisation multiobjectif et évoquerons quelques pistes de solutions. La section suivante est consacrée à la notion de hiérarchie, importante dans l'organisation des systèmes complexes mais absente des méthodes décrites dans les chapitres précédents. Une fois cette omission justifiée, nous expliquons comment un codage modulaire, répétitif et hiérarchique pourrait être associé à des gradients de sélection. La dernière section soulève le problème des biais induits par la fitness et le codage utilisé. Nous y introduisons une méthode qui pourrait permettre de les évaluer en dehors de tout problème applicatif.

## 6.1 CONTRIBUTION

NOUS AVONS PRÉSENTÉ dans cette thèse trois approches s'appuyant sur l'optimisation multiobjectif pour synthétiser des réseaux de neurones pour des tâches complexes. L'essentiel de notre contribution porte sur la mise au point de pressions de sélection adaptées aux défis de la robotique évolutionniste.

Le *bootstrap* du processus évolutionniste, c'est-à-dire l'obtention de solutions permettant d'atteindre un résultat en l'absence de gradient de sélection, est un point clef pour la synthèse de contrôleurs pour des robots. La méthode généralement utilisée est l'évolution incrémentale dans laquelle le concepteur défini des étapes intermédiaires pour guider l'évolution. Néanmoins, cette approche ne peut fonctionner que sur des exemples simples en raison des nombreuses contraintes qu'elle impose à l'évolution. Notamment, elle ne permet pas d'explorer plusieurs hypothèses à la fois pour l'ordre des sous-tâches et impose de toutes les résoudre, même si certaines sont inutiles, mal posées ou contre-productives. Nous avons commencé par montrer que l'évolution incrémentale, pouvait être vue comme un cas particulier de l'optimisation multiobjectif en considérant chaque sous-tâche comme un objectif. En employant un algorithme évolutionniste multiobjectif, nous avons pu obtenir des neurocontrôleurs pour une tâche mettant en jeu 4 étapes et plusieurs hypothèses.

Dans ces simulations, un robot mobile devait atteindre des lampes selon une séquence que l'algorithme évolutionniste devait déterminer. Malgré sa simplicité, l'approche proposée a permis de résoudre un problème parmi les plus difficiles de ceux abordés avec l'évolution incrémentale jusqu'à présent.

Nous nous sommes ensuite intéressés au maintien de la diversité dans une population en ajoutant un objectif favorisant certains individus en fonction de la population ou des individus précédemment générés. Après avoir remarqué que, dans le cas de l'évolution de réseaux de neurones et des contrôleurs en robotique mobile, il était plus intéressant de maintenir la diversité des comportements du robot ou du réseau de neurones plutôt que celle des génotypes, nous avons comparé trois approches : (1) la maximisation de la distance moyenne d'un individu avec le reste de la population, que nous appelons *diversité comportementale*, (2) la maximisation de la distance moyenne d'un individu avec une archive des comportements rencontrés, baptisée *nouveauté comportementale* et, (3) en expérience témoin, la maximisation de la distance moyenne entre les phénotypes via une distance de graphes basée sur le *graph probing*. Les objectifs de diversité et de nouveauté comportementale ont mené à d'excellents résultats pour faire évoluer un réseau de neurones avec un codage direct pour la fonction $(A \text{ xor } B)\text{et}(C \text{ xor } D)$, une fonction booléenne fortement trompeuse. Des taux de convergence proches de ceux de NEAT ont été obtenus alors que les méthodes proposées sont beaucoup plus simples et ne sont pas biaisées vers les réseaux *feed-forward*. Nous préconisons l'emploi de la diversité comportementale par rapport à la nouveauté en raison d'un coût computationnel significativement plus faible. Dans une deuxième expérience, nous avons adapté le concept de diversité comportementale à l'expérience du robot phototrope décrit dans le chapitre précédent. L'ajout de l'objectif de diversité comportementale permet d'obtenir aisément des contrôleurs efficaces malgré l'absence de gradient de sélection au début de l'évolution. De plus, le nombre de générations requis pour les obtenir est inférieur à celui nécessaire à l'approche incrémentale. L'analyse des deux méthodes montre que l'on peut voir le maintien de la diversité comportementale comme une exploration non orientée des comportements possibles alors que l'approche incrémentale multiobjectif correspond à une exploration orientée.

L'approche incrémentale introduite précédemment se concentre uniquement sur la pression de sélection, sans lien avec le génotype ou le contrôleur utilisé. Le processus évolutionniste ne peut donc s'appuyer que sur des sous-tâches directement observables à partir du comportement du phénotype. L'ingénierie adopte une démarche plus modulaire où des sous-parties sont d'abord mises au point avant d'être intégrées dans un système plus complexe. Cette approche a son pendant dans la nature via les phénomènes de symbiose et d'exaptation. Afin de mettre au point une approche modulaire et incrémentale pour la robotique évolutionniste, nous nous sommes inspirés de la littérature en biologie théorique sur l'évolution de la modularité. Nous avons ainsi pu adapter aux réseaux de neurones les concepts de parcellation, d'intégration et de dif-

férentiation, présents dans l'évolution naturelle. Ces opérations permettent de faire évoluer une correspondance génotype-phénotype modulaire. Nous nous sommes ensuite penchés sur les liens entre pression de sélection et modularité. Des modèles théoriques suggèrent qu'il est nécessaire d'aligner modules génotypiques, modules phénotypiques et gradients de selection. Dans ce travail, nous proposons de réaliser l'alignement module-phénotype en associant un objectif à chaque module phénotypique dont la fonction pourrait être utile au processus évolutionniste. Notre approche se divise en deux parties : (1) les mécanismes de parcellation, d'intégration et de différentiation, agissant sur la modularité de la correspondance génotype-phénotype et (2) un lien entre modules et objectifs, l'obtention d'un module performant constituant alors une des sous-tâches d'un processus incrémental. Cette méthode a été testée sur la fonction $(A \text{ xor } B)\text{et}(C \text{ xor } D)$. Significativement moins de générations sont requises pour obtenir des contrôleurs fonctionnels qu'avec NEAT ou l'approche basée sur la diversité du chapitre précédent. De plus, lorsque l'on ajoute un objectif de diversité comportementale, les performances sont encore nettement améliorées. L'analyse de l'ensemble des combinaisons des composants de l'approche proposée à l'aide de la valeur de Shapley montre que les éléments clefs de ces performances sont la parcellation et le multiobjectif, c'est-à-dire l'alignement entre les modules du génotype et du phénotype ainsi que celui entre les modules du phénotype et les objectifs. Cette conclusion remet en cause les approches de l'évolution de réseaux de neurones basées uniquement sur la modularité de la correspondance génotype-phénotype.

Du point de vue de l'ingéniérie automatique, nous avons proposé une méthode d'évolution de structures complexes et modulaires basée sur un point de vue original plaçant la sélection multiobjectif au coeur du processus évolutionniste. Du point de vue de l'étude des systèmes complexes, nous espérons pouvoir apporter des éléments de réponses expérimentaux concernant les liens entre génotype, sélection et modularité. Nos contributions peuvent être résumées par les point suivants :

- nous avons identifié des liens forts entre évolution incrémentale et optimisation multiobjectif ;
- nous en avons déduit une méthode simple permettant de résoudre le problème du *bootstrap* en robotique évolutionniste lorsque l'on peut définir des comportements intermédiaires ;
- nous avons montré que maintenir la diversité comportementale via l'ajout d'un objectif pouvait permettre de réduire significativement le nombre de générations requis pour converger, notamment sur des fonctions fortement déceptives ;
- en outre, cette méthode permet aussi de résoudre certains problèmes de *bootstrap* en poussant à l'exploration de nouveaux comportements ;
- nous avons proposé un génotype simple pour les réseaux de neurones exploitant les mécanismes de parcellation, d'intégration et de différentiation afin d'obtenir une correspondance génotype-phénotype modulaire ;

– nous avons proposé de faire correspondre les modules du phénotype et les gradients de sélection en s'appuyant sur l'optimisation multiobjectif ; la définition de ces nouveaux objectifs permet de fournir à l'évolution des étapes intermédiaires pour la mise au point de sous-parties du système complexe ;
– la combinaison de ce génotype et du multiobjectif ont permis d'obtenir d'excellents résultats, validant l'intérêt de la méthode proposée ;
– nos expériences ont montré que, au moins dans la tâche envisagée, il était nécessaire d'aligner modules du génotype, du phénotype et gradients de sélection pour profiter des avantages de la modularité.

Si ces contributions sont essentiellement dirigées vers l'évolution de réseaux de neurones pour le contrôle de robots, l'essentiel des conclusions et des méthodes proposées s'appliquent à la conception et l'optimisation de beaucoup de systèmes complexes avec des méthodes évolutionnistes. L'évolution incrémentale multiobjectif et la diversité comportementale ne s'appuient pas sur les caractéristiques du phénotype et peuvent donc être utilisées avec d'autres types de contrôleurs, par exemple basés sur de la logique floue (Hoffmann, 2001), ou même en soumettant à évolution la morphologie du robot (Lipson, 2006). Dans le cas de l'évolution incrémentale, la seule contrainte est d'être capable de définir des sous-tâches utiles à l'évolution. La diversité comportementale n'a de sens que si le problème traité met en jeu la notion de comportement et s'il est possible de définir une distance entre comportements. L'évolution de réseaux de neurones et la plupart des tâches de robotique mobile entrent dans cette catégorie mais on peut aussi y inclure l'ensemble des problèmes où plusieurs génotypes différents peuvent mener à des fitness égales. Le génotype proposé pour l'évolution incrémentale modulaire est conçu pour des réseaux de neurones mais il peut aisément être adapté à n'importe quel type de graphes. L'alignement entre les sous-parties d'un système complexe et des objectifs peut lui être utilisé dans n'importe quel système. Enfin, si notre conclusion portant sur l'intérêt d'aligner modules du génotype, du phénotype et gradients de sélection nécessite d'autres vérifications expérimentales, elle a vocation à être valide pour une classe de systèmes plus large que celle des réseaux de neurones.

Ces résultats permettent, à notre avis, d'avancer significativement vers la mise au point de méthodes permettant de synthétiser des systèmes complexes par évolution artificielle en exploitant la modularité et l'incrémentalité. Cependant, ils soulèvent de nouvelles difficultés que nous analyserons dans la suite de ce chapitre et auxquelles nous essayerons d'apporter quelques éléments de réponse. Nous commencerons par décrire comment l'approche proposée peut être utilisée dans certains problèmes de robotique évolutionniste afin de cerner son importance pratique. Dans la section suivante, nous aborderons les problèmes liés à l'ajout de nombreux objectifs et nous analyserons les pistes qui peuvent être suivies pour les résoudre. Nous reviendrons ensuite sur la notion de hiérarchie, dont l'importance pour l'évolvabilité a été beaucoup soulignée

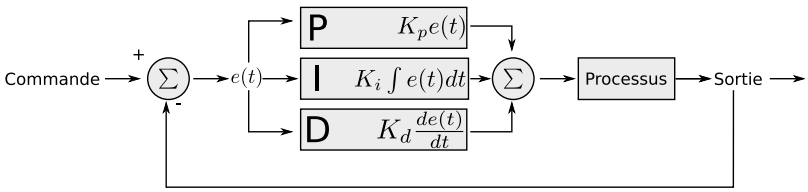



FIG. 6.1.: Diagramme blocs d'un correcteur proportionnel-intégral-dérivé (PID).

par certains auteurs (par exemple par Simon (1962)) mais qui n'est pas intégrée dans la méthode modulaire proposée. Dans une dernière section, nous examinerons la problématique des biais dans l'évolution de réseaux de neurones et plus spécialement dans le cadre des méthodes développées dans cette thèse.

## 6.2 EXEMPLES D'APPLICATION

### 6.2.1 *Introduction*

Sous un angle d'ingénierie automatique, deux grandes catégories d'applications ont été beaucoup abordées en robotique évolutionniste via l'évolution de réseaux de neurones : le pilotage de systèmes non-linéaires complexes et la locomotion rythmique. Pour chacune d'elle, nous décrivons ici des exemples d'utilisation potentielle de l'évolution incrémentale et modulaire telle que nous l'avons décrite dans le chapitre 5. Nous nous attarderons ensuite sur l'obtention de neuro-contrôleurs pour un problème de panneaux de signalisation, une tâche nécessitant l'émergence d'une mémoire.

### 6.2.2 *Pilotage*

Le pilotage de robots a été abordé en robotique évolutionniste dans de nombreux travaux, essentiellement en simulation. On peut citer, par exemple, le contrôle d'un dirigeable lenticulaire (Doncieux et Meyer, 2003b), de planeurs autonomes exploitant les ascendances thermiques et dynamiques (Doncieux et al., 2007) ou les gradients de vent (Barate et al., 2006), une fusée (Gomez et al., 2006) et des pendules inversés (Stanley et Miikkulainen, 2002a; Geva et Sitte, 1993; Doncieux et Meyer, 2005; Gomez et al., 2006; Igel, 2003). Dans toutes ces situations, il s'agit de trouver des contrôleurs réactifs prenant en compte des capteurs pour asservir une machine aux dynamiques complexes, souvent non-linéaires et mal comprises. Ces méthodes visent un champ d'application voisin de celles développées dans le domaine de l'automatique (voir Levine (1996) pour un livre de référence), leurs différences se situant essentiellement au niveau de la quantité de connaissances du système requises de la

part de l'expérimentateur. Les automaticiens mettent généralement au point des boucles de contrôle s'appuyant sur des modules plus simples dont l'agencement est décrit par des diagramme blocs (figure 6.1). Parmi les modules courants, on peut citer les correcteurs proportionnel (module P), dérivée (module D) et intégrale (module I), mais des fonctions plus complexes, par exemple un filtre de Kalman, peuvent constituer un bloc. Cette conception modulaire se marie particulièrement bien avec l'approche modulaire que nous avons proposée.

Le pilotage d'un drone à voilure fixe, par exemple, nécessite le maintien des angles de roulis, lacet et tangage à l'aide des ailerons, de la dérive et de la profondeur avec potentiellement des couplages dûs entre autres aux effets de lacet inverse. Un neuro-contrôleur pilotant un tel drone possédera donc trois entrées, trois sorties et une structure inconnue mais raisonnablement complexe. Il y a cependant de fortes chances que le pilotage efficace nécessite l'utilisation de modules PD (proportionnel-dérivée) ou même PID. En suivant la méthode proposée, il est possible d'aider le processus évolutionniste en ajoutant un objectif correspondant aux performances des sous-réseaux pour la fonction PID et en utilisant le génotype modulaire. Ainsi, l'évolution pourra exploiter autant de fois que nécessaire un bloc PID et ainsi potentiellement obtenir des contrôleurs performants pour le drone à voilure fixe. Le pilotage d'un hélicoptère peut être obtenu de manière similaire en combinant différents modules PID reliés aux différentes commandes (De Nardi et al., 2006). Comparée à une approche classique fondée sur l'automatique, la méthode évolutionniste incrémentale modulaire requiert moins de connaissance : l'évolution choisit quels modules sont nécessaires, comment les agencer et comment les régler ; chaque sous-objectif constitue alors uniquement une suggestion. Par rapport à une approche évolutionniste directe, sans sous-tâches ni modules, les chances d'obtenir des contrôleurs aux performances comparables avec celles obtenues en automatique sont augmentées.

### 6.2.3 *Locomotion rythmique*

La locomotion animale est caractérisée par une activité rythmique et l'utilisation de nombreux degrés de liberté. L'ondulation d'un serpent, le trot d'un cheval ou le vol d'un oiseau ont en commun la présence d'oscillations que le système nerveux ajuste à chaque instant. N'importe quel insecte est expert dans le maniement de systèmes possédant de nombreux degrés de liberté couplés entre eux et plongés dans un environnement complexe. Ces performances, difficiles à égaler avec un robot, s'appuient sur des circuits neuronaux oscillants particuliers, appelés *Central Pattern Generators* (CPGs). Les capacités locomotrices des êtres vivants ont inspirés de nombreux travaux en robotique évolutionniste autour de la marche (Lewis et al., 1993; Gallagher et al., 1996; Jakobi, 1998; Kodjabachian et Meyer, 1997), du vol (Mouret et al., 2006) et de la nage (Ijspeert et al., 1999).

Parmi ces travaux, Ijspeert et al. (1999) a suivi une approche incrémentale basée sur une décomposition en modules pour obtenir des contrôleurs pour une lamproie simulée. Contrairement à la méthode que nous avons présentée dans cette thèse, l'assemblage des modules et l'ordre des étapes n'a pas été laissé au processus évolutionniste. Dans un premier temps, un oscillateur dont la fréquence peut varier en fonction d'un stimulus externe a été évolué. Cet oscillateur a ensuite été recopié plusieurs fois de façon à ce qu'un oscillateur soit connecté aux muscles de chaque segment de la lamproie. Les connections reliant chaque oscillateur, déterminant leur déphasage, ont ensuite été soumises à évolution pour obtenir des réseaux de neurones capables de contrôler la lamproie simulée en eaux calmes. Dans une troisième étape, des capteurs d'étirement des muscles ont été ajoutés et connectés par l'évolution au reste du réseau afin de permettre à la lamproie de traverser une barre d'eau agitée.

Cette division en étapes et en modules peut être reproduite avec la méthode d'évolution modulaire incrémentale que nous avons proposée, ce qui permettrait de relâcher certaines contraintes comme la génération de passage à l'étape suivante et la manière de recopier et connecter les oscillateurs segmentaux. De plus, notre méthode permettrait d'utiliser directement les capteurs si cela s'avère plus simple pour l'évolution. Pour mettre en place notre approche, nous devons définir trois objectifs : (1) la capacité d'un module à osciller régulièrement et à varier en fonction d'un stimulus ; (2) la capacité de la lamproie à se mouvoir en eaux calme ; (3) la capacité de la lamproie à nager en eaux agitées. On peut s'attendre à ce que des comportements simples mais moins efficaces que l'ondulation, par exemple n'utiliser qu'un seul muscle pour « battre de la queue », dominent rapidement la population. Pour éviter ce problème, il est possible de définir un objectif de diversité comportementale en décrivant chaque comportement, par exemple, par un vecteur de dimension le nombre de muscles et dont chaque composante contiendrait le nombre de pas de temps où chaque muscle a été contracté. Ainsi, une lamproie n'utilisant que certains de ses muscles sera très différente d'une les utilisant tous et l'objectif de diversité devrait pouvoir inciter à l'exploration d'ondulations complexes. Si on peut raisonnablement espérer obtenir des contrôleurs fonctionels en suivant cette méthode, il faut noter que la morphologie de la lamproie n'est pas prise en compte. La dissection d'une lamproie révèle notamment qu'à chaque segment de la lamproie est associé un oscillateur et que ceux-ci sont connectés entre eux de proche en proche. Ces informations, qui réduisent beaucoup l'espace de recherche ne peuvent pas être prises en compte par notre méthode. Nous reviendrons dans la suite du chapitre de discussion sur l'utilisation de biais topologiques dans l'évolution.

Une démarche utilisant des étapes et des modules a aussi été utilisée pour synthétiser un réseau de neurones capable de contrôler un robot hexapode capable de marcher, éviter les obstacles et suivre un gradient (Kodjabachian et Meyer, 1997). Durant la première étape, un réseau de neurones permettant à un insecte simulé de marcher a été évolué. À la fin de cette étape, le meilleur

individu est sélectionné pour constituer le contrôleur de locomotion utilisé par la suite. Dans les étapes suivantes, d'autres modules sont évolués pour permettre le suivi de gradient puis l'évitement d'obstacles. Ces nouveaux modules peuvent influencer le comportement du module de locomotion en créant des connexions inter-modules. Comme nous l'avons proposé pour la lamproie, il est là aussi aisé de transposer ce travail vers notre approche afin d'assouplir les contraintes imposées par l'ordre et le critère de changement des étapes. Trois objectifs peuvent être définis : (1) la capacité de l'insecte à se déplacer ; (2) la capacité à éviter les obstacles ; (3) la capacité à suivre un gradient. Ces trois objectifs peuvent être évalués uniquement en analysant le comportement du robot, permettant de suivre une approche sans prise en compte de la modularité (chapitre 3). Néanmoins, il est probable qu'une approche modulaire se révèle plus performante. On peut alors envisager de tester (1) la capacité d'un module à contrôler les pattes pour la marche ; (2) la capacité d'un module à générer une valeur positive lorsqu'il y a un obstacle à gauche, négative à droite, et nulle sinon ; (3) la capacité d'un module à effectuer un comportement similaire pour suivre un gradient. Ces gradients de sélection peuvent sembler imposer des contraintes très fortes sur les modules à obtenir mais il faut noter que notre approche multi-objectif permet d'ignorer un objectif s'il est mal spécifié. Ainsi, s'il est plus simple de ne pas utiliser l'un des objectifs ou s'il s'avère plus facile de savoir déjà éviter les obstacles avant de suivre un gradient, c'est-à-dire de suivre l'ordre inverse de celui suivi par Kodjabachian et Meyer (1997), l'évolution pourra le faire.

Une approche similaire peut aussi être mise en place pour faire évoluer des contrôleurs de battement d'ailes pour un robot à ailes battantes comme nous l'avons fait précédemment (Mouret et al., 2006). On pourra définir (1) un objectif récompensant la présence d'un module oscillant, (2) un objectif récompensant le vol en ligne droite (3) un objectif récompensant le vol en virage. Un vecteur de comportement comportant, par exemple, la fréquence de battement observée pour chaque aile et l'amplitude minimum et maximum utilisée pour chaque degré de liberté pourrait être utilisée pour ajouter un objectif de diversité comportementale. Un tel objectif pourrait permettre de garantir qu'un large spectre de types de battements sont explorés et, notamment, d'éviter la convergence vers des comportements de planeurs, très stables et robustes mais insuffisants pour maintenir une altitude constante pendant une longue période.

### 6.2.4 *Exemple détaillé*

Nous détaillons ici l'application de la méthode du chapitre 5 à un problème de panneaux de signalisation (*road-sign*) pour un robot mobile (section 2.5.1). Comme nous l'avons expliqué précédemment, ce type de problèmes constitue un exemple simple de comportement requérant l'émergence et l'utilisation d'une mémoire. Un robot doit effectuer un choix de direction en fonction

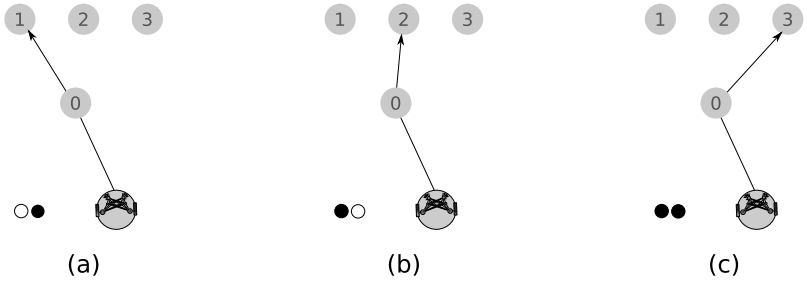



FIG. 6.2.: Problème de panneau de signalisation utilisé. Un robot mobile est équipé de quatre paire de capteurs directionels, chacune sensible à une zone particulière (0, 1, 2 ou 3). Pendant les dix premiers pas de temps de l'expérience, il reçoit une information de direction sous la forme de deux bits (ronds noirs et blancs sur la figure). Il doit ensuite se diriger vers la zone 0 puis choisir d'aller vers la zone 1,2 ou 3 en fonction de l'information reçue au début de l'expérience. Ce problème nécessite l'utilisation d'une mémoire et le décodage du type de la zone cible.

d'une information donnée longtemps avant ; il doit donc posséder une mémoire rudimentaire lui permettant de retenir la direction et être capable de l'employer pour effectuer le choix. De nombreuses variantes de complexité diverses peuvent être définies en modifiant le nombre de directions et la manière de l'indiquer au robot.

Dans cette section, nous nous intéressons à une variante où le robot doit choisir parmi trois directions après avoir reçu une information sous la forme de deux bits (figure 6.2). Quatre zones colorées sont disposées dans une arène simulée et le robot dispose de quatre paires de capteurs directionnels renvoyant chacun 1 si la zone colorée associée est dans le champ de vision et 0 sinon. Le robot reçoit pendant les dix premiers pas de temps d'une expérience deux bits d'information, par exemple sous la forme de lumières, qu'il perçoit par deux capteurs. Il doit ensuite se rendre vers la zone 0 puis choisir sa direction, 01 correspondant à la zone 1, 10 à la 2 et 11 à la 3. L'essentiel de la difficulté réside dans l'interprétation puis la mémorisation des bits de direction. Il est en effet relativement aisé de trouver par évolution des réseaux de neurones capables de résoudre un problème semblable mais comportant seulement deux zones. Dans ce cas, un simple neurone récurent connecté au capteurs de direction peut mémoriser la direction, qui peut être exploitée lorsque la zone 0 est atteinte. Lorsqu'une troisième zone est ajoutée, l'évolution atteint aisément le même point, ce qui lui permet de rejoindre les zones 1 et 2 et d'améliorer significativement la fitness des meilleurs individus. Néanmoins, de profonds changements sont nécessaires pour changer le mécanisme de mémoire pour atteindre la troisième zone car un décodeur capable d'interpréter les deux bits doit être ajouté et son résultat mémorisé à la place des données d'entrée. Si on

utilise une approche évolutionniste classique, avec une unique fitness, l'apparition de ce « décodeur-mémorisateur » n'amène à aucun avantage sélectif tant que la bonne circuiterie n'est pas connectée pour qu'il permette de rejoindre la zone 3. Un tel module a donc très peu de chances d'apparaître.

Il est possible d'appliquer la méthode d'évolution incrémentale modulaire développée dans le chapitre 5 afin d'ajouter un gradient de sélection pour les modules décodeur-mémorisateurs. Ainsi, le processus évolutionniste devrait pouvoir obtenir ces modules à deux entrées et trois sorties puis les utiliser pour contrôler le robot mobile. Nous présenterons ensuite des résultats préliminaires, qui seront approfondis dans de futurs travaux.

Afin d'éviter le sur-apprentissage, le robot doit être testé pour les trois cas à partir d'au moins deux positions différentes, à chaque fois pour au moins deux arrangements de zones. Au total, douze expériences sont donc requises au minimum pour évaluer la fitness. L'évaluation des modules décodeur-mémorisateur peut se faire « *in vivo* », c'est-à-dire en observant la valeur des sorties des modules pendant l'évaluation du comportement du robot, ou « *in vitro* » en testant les modules séparément pour les trois cas d'entrées possibles. Cette dernière solution est moins contraignante pour le processus évolutionniste, la seule information donnée étant qu'un tel module peut être utile, mais elle complique aussi beaucoup la tâche de l'évolution qui doit en plus réussir à connecter le module aux bonnes entrées du réseau. En suivant une approche « *in vivo* », on peut augmenter la fitness d'un module lorsque la sortie $n$ ($n = 1, 2, 3$) prend la valeur 1 pendant la durée de l'expérience si la zone cible est $n$ et 0 sinon. Cette approche a comparativement plus de chances de succès car la pression de sélection s'exerce à la fois sur les performances du module mais aussi sur sa connexion aux entrées. Enfin, il est possible d'ajouter un objectif de diversité comportementale, le chapitre 4 ayant montré l'intérêt pratique d'une telle démarche.

Suite à cette analyse, nous proposons d'utiliser une fitness composée de trois objectifs à maximiser correspondant respectivement à (1) $F_z$, l'opposé du nombre de pas de temps pour atteindre la zone cible dans chacune des situations (2) $F_{dec}$, le meilleur score des modules pour la sous-tâche et (3) $F_{div}$, la diversité comportementale. $F_{dec}$ peut être évaluée « in vivo »[1] en réalisant la somme des différences entre la valeur attendue et la valeur obtenue pour les modules compatibles (deux entrées, trois sorties). En s'inspirant de la section 4.3.2, on peut associer à chaque individu $i$ un vecteur booléen $\mathbf{v}^{(i)}$ de comportements décrivant les zones atteintes dans chacun des tests. La distance comportementale $d(i, j)$ entre les individus $i$ et $j$ peut alors être définie comme la distance euclidienne entre $\mathbf{v}^{(i)}$ et $\mathbf{v}^{(j)}$.

Afin d'obtenir des résultats préliminaires, l'algorithme NSGA-II a été lancée 4 fois avec une population de 200 individus et les paramètres de l'annexe

1 L'utilisation d'une évaluation « *in vitro* » dans laquelle le module est soumis à un jeu d'entrées particulier est possible mais nécessite la connexion du module obtenu aux bonnes entrées. La tâche demandée à l'évolution est alors plus complexe qu'en utilisant une approche « *in vivo* », plus explicite.

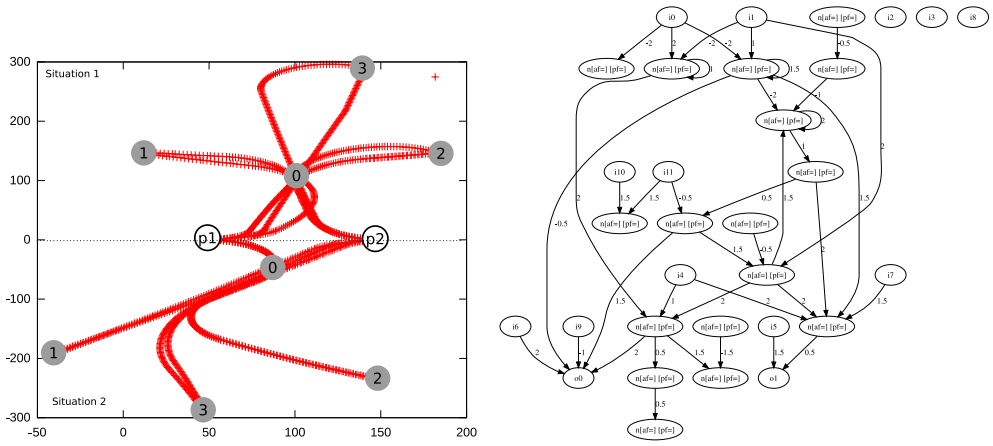



FIG. 6.3.: (a) Trajectoires obtenue dans les différentes situations pour le problème de panneaux de signalisation. En fonction des informations reçues, le robot prend toujours la bonne décision. (b) Réseau de neurones obtenu par évolution contrôlant le robot sur les trajectoires (a).

A.2.3. En 150000 générations, les 4 expériences lancées ont permis de trouver des contrôleurs capables de résoudre la tâche dans toutes les situations. Dans les 4 cas, les individus avec la meilleure fitness $F_r$ ont une bonne fitness $F_{div}$, ce qui suggère qu'un sous-module capable de réaliser la tâche décodeur-memorisateur est exploité. Ce nombre de génération très important peut certainement être amélioré et doit surtout être comparé à une expérience témoin sans objectif pour les sous-modules. La figure 6.3 présente un ensemble de trajectoires obtenues pour les différentes situations et le réseau de neurones correspondant.

## 6.3 OPTIMISATION DE NOMBREUX OBJECTIFS

### 6.3.1 *Introduction*

Le nombre très important de générations requises pour obtenir des contrôleurs dans le cas des panneaux de signalisation peut en partie s'expliquer par le fait qu'en quelques centaines de générations l'ensemble de la population se situe sur le front de Pareto (figure 6.4). La pression de sélection est alors réduite au maintien de la diversité sur le front exercé par NSGA-II via la distance de peuplement (*crowding distance*). Cette pression peut se révéler insuffisante pour faire émerger des individus performants. Au moins deux situations non exclusives sont susceptibles d'être à l'origine de cette pathologie :

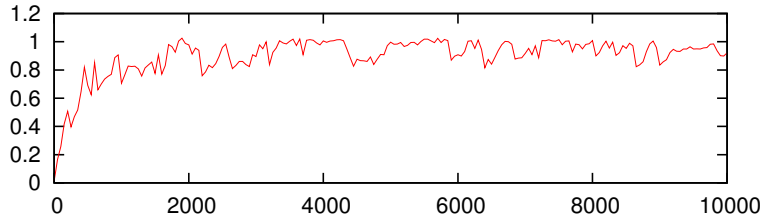



FIG. 6.4.: Proportion d'individus sur le front de Pareto dans l'expérience de panneaux de signalisations de la section 6.2.4. En quelques centaines de générations, plus de 80% de la population est non-dominée.


(a) (b)

FIG. 6.5.: (a) Dans un problème bi-objectif, la relation de Pareto entre un individu $I$ et le reste de la population divise l'espace objectif en trois ensembles : les individus dominant $I$ (B), les individus dominés par $I$ (W), et les individus équivalents (E). (b) Dans un problèmes tri-objectif, la proportion des solutions équivalentes augmente.

1. la population a atteint un maximum (local ou global) et donc n'arrive pas à progresser ; dans ce cas, le front se rempli de copies des mêmes solutions ;
2. le nombre de compromis possible est suffisamment grand pour qu'il soit aisé de générer une infinité de solutions non-dominées.

La première situation se rencontre essentiellement lorsque les objectifs ont des valeurs discrètes, l'égalité de variables réelles étant difficile à obtenir. L'ajout d'un objectif de diversité comportementale et les méthodes de maintien de la diversité sur le front de Pareto peuvent exercer une pression efficace vers l'exploration pour s'échapper du maximum local, comme nous l'avons vu dans le chapitre 4. L'utilisation d'une relation d'ordre différente de la domination de Pareto peut aussi avoir une influence sur ce problème ; nous aborderons cette piste après avoir analysé la deuxième situation.

La deuxième situation repose sur les caractéristiques de la relation de domination de Pareto. En définissant un ordre partiel, celle-ci divise l'espace objectifs en trois ensembles : les solutions meilleures, moins bonnes et équivalentes. La figure 6.5 illustre la division de l'espace pour les problèmes à deux et trois objectifs. Plus généralement, si l'on désigne par $e$ la portion d'un espace ob-

jectifs $M$-dimensionel contenant tous les vecteurs objectifs classés comme équivalents à un vecteur donné, alors $e$ augmente en fonction de $M$ selon l'équation (Farina et Amato, 2004) :

$$e = \frac{2^M - 2}{2^M}$$

Quand $M$ tend vers l'infini, $e$ tend vers 1, c'est-à-dire vers l'ensemble de l'espace. Par conséquent, plus le nombre d'objectifs augmente, plus un individu quelconque a de chances d'être non-dominé.

Cette propriété, et plus généralement les mauvaises performances des algorithmes évolutionnistes multi-objectifs lorsque le nombre d'objectifs augmente, ont été observées expérimentalement pour l'ensemble des algorithmes multiobjectifs directement basés sur la relation de domination de Pareto. Purshouse et Fleming (2007) ont mentionné l'inefficacité de NSGA-II lorsque plus de trois objectifs sont utilisés, même quand une grande population est utilisée. Leurs expériences ont aussi souligné la grande influence de la méthode de maintien de la diversité sur le front de Pareto dans ces situations. Ces conclusions rejoignent celles de Khare et al. (2003) qui ont comparé NSGA-II, SPEA2 et PESA sur des problèmes comportant de deux à huit objectifs. Bien qu'en terme de convergence vers le front de Pareto, PESA obtienne les meilleurs performances, cet algorithme s'appuie sur des calculs computationnellement trop coûteux pour être utilisable au-delà de quelques objectifs. Dans d'autres travaux Wagner et al. (2006) concluent à la supériorité de $\varepsilon$-MOEA sur NSGA-II et SPEA-2 mais constatent eux aussi la dégradation des performances liée à la hausse du nombre d'objectifs. Les observations empiriques sur le comportement des algorithmes multiobjectifs en présence de nombreux objectifs sont soutenues par l'étude théorique de Teytaud (2006) dans laquelle il est montré que, à moins que le nombre d'objectifs en conflit soit très modéré (typiquement inférieur ou égal à trois), n'importe quel algorithme multiobjectif basé sur des comparaisons ne sera pas vraiment plus performant qu'une recherche aléatoire.

Les difficultés que rencontrent les algorithmes évolutionnistes multiobjectif dans cette situation touchent particulièrement les méthodes proposées dans cette thèse. Un objectif est ajouté pour chaque étape ou hypothèse dans l'évolution incrémentale multiobjectifs (chapitre 3). Suivant cette approche, nous sommes donc amenés à résoudre des problèmes pouvant comporter plus de 10 objectifs, nombre augmentant si les tâches abordées sont plus difficiles. La situation est similaire pour l'évolution incrémentale modulaire (chapitre 5). Puisque nous proposons d'associer à chaque module un objectif, le nombre d'objectifs croît avec la complexité des tâches. Le nombre d'objectifs est moins critique dans le cas de la diversité comportementale (chapitre 4) car un seul objectif est ajouté. Cependant, les expériences montrent que l'ajout d'un objectif n'est généralement pas anodin et peut se traduire par une dégradation des performances.

### 6.3.2 *Exploitation de la structure incrémentale*

L'influence des problèmes de performances des algorithmes multiobjectif sur les méthodes proposées peut cependant être tempéré en notant qu'elles induise un espace objectif très particulier dans lequel tous les objectifs n'ont pas la même utilité à toutes les générations. D'un point de vue théorique et en rapport avec les résultats de Teytaud (2006), les objectifs utilisés dans les méthodes incrémentales de cette thèse ne sont en conflit que lors de l'exploration simultanée de plusieurs hypothèses ou lorsque des individus sur-spécialisés peuvent apparaître. Les méthodes proposées ne sont donc pas soumises aux mêmes bornes minimales de complexité qu'une optimisation multiobjectif quelconque.

Si le problème possède une structure idéale simple, le processus évolutionniste devrait commencer par optimiser les individus en appliquant la pression sélective correspondant à la première étape. Puis, dès que l'étape suivante pourra être testée, il optimisera simultanément les deux objectifs jusqu'à obtenir la fitness maximum pour le premier objectif, déplaçant alors la pression de sélection sur le second, à laquelle s'ajoutera dès que possible celle sur le troisième. Dans ce cas simple, l'objectif correspondant à l'étape courante et celui associé à l'étape suivante fournissent alors l'essentiel de la pression de sélection. Le front de Pareto s'est alors d'abord déplacé selon le premier objectif, puis les deux premiers, puis le deuxième et le troisième, etc... Dans une situation réelle, le recouvrement entre les objectifs peut être plus important mais, dans un problème incrémental, une partie des objectifs correspondant aux anciennes étapes et ceux associés aux étapes dans un futur lointain jouent un rôle moins importants que les autres dans la pression de sélection. Pour réduire le nombre d'objectifs, c'est donc vers ces derniers que nous devons nous tourner. Dans le cas, plus complexe, où plusieurs hypothèses sont évaluées simultanément, plus d'objectifs sont nécessaires mais, là aussi, les objectifs correspondant aux premières étapes sont moins importants quand l'évolution passe d'étapes en étapes.

En faisant l'hypothèse que la tâche abordée possède une structure incrémentale, on peut donc isoler à chaque génération un sous-ensemble des objectifs qui représentent le mieux les déplacements du front de Pareto. Ces objectifs sont ceux qui exercent l'essentiel de la pression de sélection, il est donc envisageable de n'utiliser qu'eux lors de la phase de sélection. À l'opposé, les objectifs selon lesquels aucun progrès n'est fait pendant plusieurs générations, soit parce qu'ils sont trop difficiles soit parce qu'ils sont déjà optimisés, peuvent être ignorés. Pour estimer les déplacements du front de Pareto et peut-être aussi pour conserver une archive, il est cependant nécessaire de continuer à calculer le front de Pareto sur l'ensemble des objectifs.

Plus concrètement, si on souhaite sélectionner $N$ objectifs parmi $M$ et que l'on note par $P_g$ le front de Pareto à la génération $g$, on peut calculer la variance

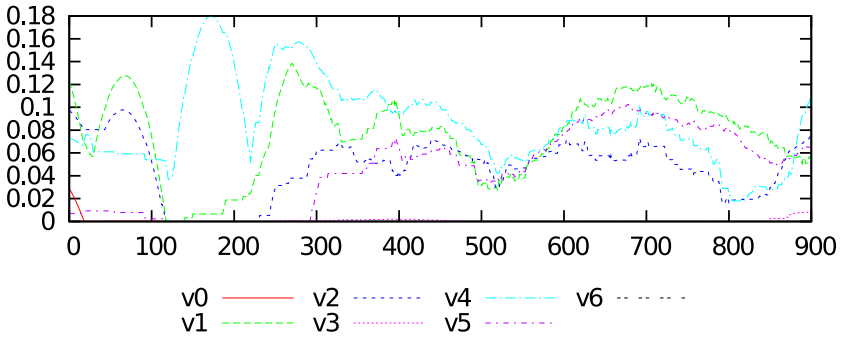



FIG. 6.6.: Variance $v_i^{(g)}$ des individus du front de Pareto observée dans une expérience typique de robot phototrope, pour chacun des 7 objectifs ($k = 100$). Suivant les générations, le front ne s'est pas déplacé autant selon tous les objectifs.

$v_i^{(g)}$ de la fitness de l'ensemble des individus de $P = P_g \bigcup P_{g-1} \bigcup \cdots P_{g-k}$ pour chaque objectif $i$ pendant les $k$ dernières générations :

$$\begin{aligned} \bar{x_i}^{(g)} &= \frac{1}{k \cdot |P_g|} \sum_{i=(g-k)}^{i=g} \sum_{x \in P_g} x_i \\ v_i^{(g)} &= \frac{1}{k \cdot |P_g|} \sum_{i=(g-k)}^{i=g} \sum_{x \in P_g} (x_i - \bar{x_i}^{(g)})^2 \end{aligned}$$

À la génération $g$, on peut alors choisir les $N$ objectifs qui ont la plus grande variance $v_i^{(g)}$. La figure 6.6 montre les valeurs $v_i^{(g)}$ pour une expériences de robot phototrope avec les objectifs de la section 3.3 et $k = 100$. On peut y comprendre la dynamique suivie par le processus évolutionniste : pendant les 100 premières générations, les individus ont surtout progressé pour atteindre les lampes 1 et 2 ; ils ont ensuite réduit le temps pour attendre la lampe 4 ; au bout de 500 générations, la progression s'est aussi faite selon l'objectif de la lampe 5. La lampe 0, facile à atteindre et indispensable pour la suite, a une valeur $v_i^{(g)}$ nulle pendant l'essentiel de l'expérience. En fixant $N = 2$, dans ces différentes situations, la valeur de $v_i^{(g)}$ nous aurait amené à choisir d'abord les objectifs 1 et 2, puis 1 et 4 et, enfin 4 et 5. La valeur $v_i^{(g)}$ semble donc bien refléter la dynamique de l'évolution incrémentale dans cette expérience. Il faut néanmoins insister sur le fait que l'utilisation d'une autre pression de sélection peut mener à des dynamiques très différentes. Par conséquent, en réduisant le nombre d'objectifs comme nous le proposons, nous obtiendrons certainement une figure très différente. D'une manière similaire aux expériences de co-évolution, il est aussi possible que l'on observe des cycles.

D'autres approches mettent en jeu de nombreux objectifs et sont donc confrontées aux même problèmes que celles que nous proposons. La co-évolution de Pareto (de Jong et Pollack (2004a), voir la section 2.4.3) est une méthode pour traiter la classe de problèmes d'optimisation où la qualité des solutions candidates est estimée par leur performance sur un certain nombre de tests. Une population de tests co-évolue avec celle des solutions candidates et ces dernières sont ordonnées à l'aide d'un classement de Pareto où la performance pour chaque test constitue un objectif. Dans ce contexte, de très nombreux objectifs sont créés, ce qui rend l'utilisation de cette approche peu utilisable en pratique avec les méthodes d'optimisation multiobjectif actuelles. Afin de limiter ce problème, Bucci et al. (2004) proposent de placer les tests selon le nombre minimum d'axes de coordonnées qui préservent l'ordre parmi les solutions. Les auteurs montrent qu'il est ainsi possible de réduire le nombre de dimensions en trouvant les objectifs sous-jacents au problème. Un algorithme polynomial sans garantie de trouver les axes de coordonnées minimaux a été publié (Bucci et al., 2004).

### 6.3.3 *Modification la relation de domination*

Une alternative à la réduction du nombre d'objectifs est de modifier la relation de Pareto afin de concentrer la recherche sur un sous-ensemble des solutions non-dominées (Farina et Amato, 2004; Praditwong et Yao, 2007; Corne et Knowles, 2007). En plus de réduire les difficultées liées au trop grand nombre de solutions non dominées possibles, l'utilisation d'une relation de domination modifiée pourrait aussi avoir une influence sur la première situation décrite au début de cette section, où le processus évolutionniste est bloqué dans un extremum local. Selon Farina et Amato (2004), la relation de Pareto est peu satisfaisante pour deux raisons :

1. le nombre d'objectifs pour lesquels un individu $x^{(1)}$ obtient un meilleur score qu'un individu $x^{(2)}$, $x^{(1)}$ et $x^{(2)}$ étant non-dominés, n'est pas prise en compte ;
2. la valeur (normalisée) des améliorations pour chaque objectif est négligée.

Par exemple, un individu dominant l'ensemble de la population pour tous les objectifs sauf un est considéré comme étant équivalent à un autre ne dominant que sur un objectif. La $k$-optimalité (Farina et Amato, 2004) permet de rapprocher la notion de domination de l'intuition en posant qu'un individu $x^{(1)}$ $k$-domine un individu $x^{(2)}$ si et seulement si :

$$\begin{cases} n_e < M \\ n_b \geq \frac{M-n_e}{k+1} \end{cases}$$

où $n_e$ désigne le nombre d'objectifs pour lesquels $x^{(1)}$ et $x^{(2)}$ sont égaux et $n_b$ le nombre de ceux pour lesquels $x^{(1)}$ est meilleur que $x^{(2)}$. Farina et Amato

(2004) proposent ensuite d'étendre cette définition en utilisant de l'arithmétique floue pour décrire le nombre d'objectifs améliorés entre deux individus et pour prendre en compte la valeur des amélioration. Cette définition n'a, à notre connaissance, pas encore été utilisée pour créer de nouvelles méthodes d'optimisation multiobjectif. Une autre modification de la relation de domination a été cependant testée avec une version modifiée de NSGA-II pour six objectifs (Praditwong et Yao, 2007). Les résultats ont révélé une amélioration significative de la convergence vers le front de Pareto, soulignant le potentiel d'une telle approche.

## 6.4 HIÉRARCHIE

### 6.4.1 *Importance de la hiérarchie*

Les organismes vivants et les artefacts complexes montrent tous une évidente organisation hiérarchique, les modules de chaque niveau d'abstraction étant combinés ensemble en des modules plus abstraits. Le corps humain, par exemple, peut ainsi être décomposé en organes, puis en cellules, chacune d'elles pouvant à leur tour être décrites comme l'assemblage d'un noyau, de cytoplasmes et de nombreux organites, etc. L'importance de la hiérarchie amène Simon (1962) à en faire l'un des principes généraux de l'organisation des systèmes complexes, vivants ou artificiels. Comme nous l'avons souligné dans la section 2.2.2, la composition hiérarchique de modules est l'une des clefs pour la conception mais aussi l'évolution de ces systèmes. Malgré cette importance, ce concept est presque absent de la méthode d'évolution modulaire incrémentale présentée dans cette thèse.

La raison principale de cette omission est notre volonté de simplicité. On peut, en effet, s'interroger de la pertinence de la hiérarchie dans les problèmes accessibles aux méthodes évolutionnistes actuelles. Comme l'a souligné Watson (2005), mettre au point une tâche requérant des solutions modulaires, hiérarchiques et répétitives tout en étant suffisamment simple pour être analysée est un défi difficile, qui, paradoxalement, nécessite une connaissance profonde de l'évolution des systèmes complexes. Pour des raisons de temps de calcul mais aussi méthodologiques, les problèmes sur lesquels il est envisageable de tester de nouvelles approches pour l'évolution artificielle mettent rarement en jeu plus d'une vingtaine de neurones, le plus souvent utilisés avec moins d'une dizaine de capteurs et d'actionneurs. Dans des systèmes si simples, il est difficile d'envisager plus de quelques niveaux de hiérarchie. L'approche modulaire que nous avons présentée dans le chapitre 5 implémente un unique niveau de hiérarchie, le réseau principal étant composé de modules non décomposables. Il est probable que cet unique niveau suffise pour améliorer les performances des méthodes évolutionnistes pour de nombreuses tâches encore difficilement abordables.

Nous avons écarté la hiérarchie de la correspondance génotype-phénotype utilisée de cette thèse car nous souhaitions nous concentrer sur les liens entre modules et sélection. Cependant, l'utilisation d'autres codages modulaires peut être envisagée avec l'approche multiobjectif du chapitre 5 à partir du moment où l'alignement génotype-phénotype-sélection est conservé. Nous avons proposé dans d'autres travaux un codage modulaire, régulier et hiérarchique (Mouret et Doncieux, à paraître), qui pourrait être employé. Après l'avoir rapidement décrit, nous montrerons comment l'intégrer avec la méthode proposée dans cette thèse.

### 6.4.2 *Le codage MENNAG*

Un système hiérarchique se représente naturellement sous la forme d'un arbre, les feuilles en constituant les éléments les plus élémentaires et chaque nœud un module. Partant de ce principe, il est aisé de rapprocher un arbre décrivant une structure avec un programme informatique, lui aussi efficacement représenté par le même type de structure arborescente. De très nombreux outils ont été développés pour faciliter la description et la formalisation des langages informatiques, parmi lesquels les grammaires hors-contexte sont indubitablement les plus utilisés. Il semble donc naturel d'essayer de définir une grammaire hors-contexte pour la description des systèmes modulaires. Une telle démarche a été notamment suivie dans des travaux antérieurs pour l'évolution de réseaux de neurones par codage cellulaire (Gruau, 1995) et le paradigme SGOCE (Kodjabachian et Meyer, 1997). De telles grammaires définissent la syntaxe d'un langage mais pas sa sémantique, elles ne suffisent donc pas à le décrire complètement. Les grammaires à attributs (Knuth, 1968; Paakki, 1995) les étendent en ajoutant à chaque symbole des attributs et à chaque règle des calculs de mise à jour de ces attributs. L'évaluation d'un programme revient alors à évaluer l'ensemble des attributs liés aux symboles dont il est constitué. Hussain (2003) a pu montrer que de nombreux codages de réseaux de neurones pouvaient être formalisés sous forme de grammaires à attributs spécifiant des programmes évolués en utilisant les opérateurs de la programmation génétique. Poursuivant ce concept, nous avons donc défini MENNAG (Modular Encoding for Neural Network based on Attribute Grammars, Mouret et Doncieux (à paraître)), un codage modulaire indirect formalisé sous forme d'une grammaire à attributs.

La figure 6.7 résume le fonctionnement de MENNAG : la grammaire est utilisée pour générer des arbres aléatoires syntaxiquement corrects et pour les opérateurs génétiques ; les attributs sont ensuite évalués pour obtenir une liste de neurones, identifiés par un nombre binaire reflétant leur position dans l'arbre, et une liste de connexions ; un réseau de neurones est construit à partir de ces deux listes. La structure principale de la grammaire de MENNAG repose sur les non-terminaux DIV et CELL permettant respectivement de créer un nouveau module, en créant un nouveau niveau dans l'arbre, et de créer un

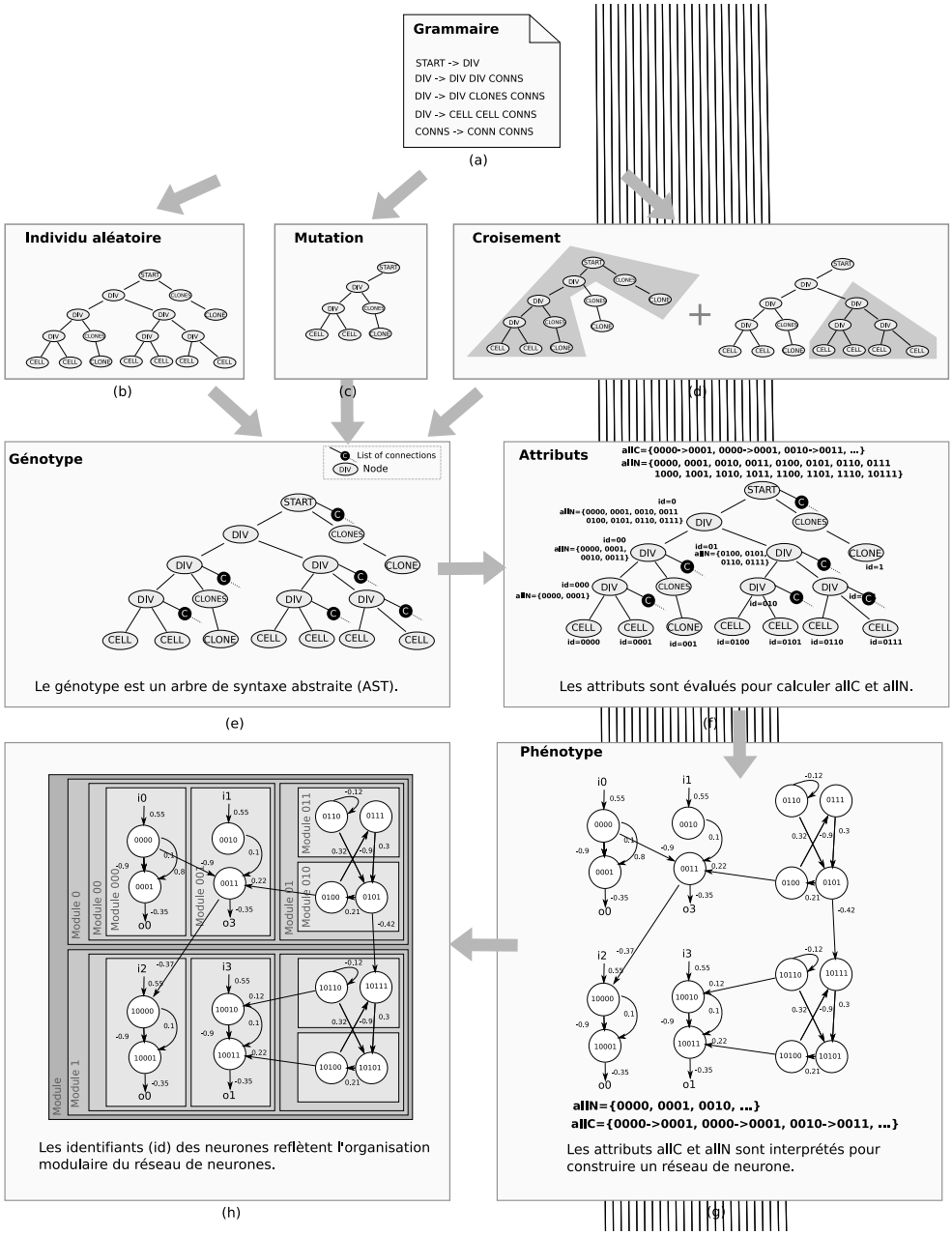



FIG. 6.7.: La grammaire à attributs (a) est utilisée par les opérateurs génétiques ((b), (c), (d)) pour créer de nouveaux génotypes syntaxiquement corrects, définis comme des arbres de syntaxe (e). Les attributs de la racine de l'arbre *allC* (toutes les connections) et *allN* (tous les neurones) sont ensuite évalués (f). Ces attributs sont ensuite facilement transformés en un réseau de neurones (g) en parcourant les deux listes. Enfin, le phénotype peut être divisé en modules (h) en utilisant les identifiants des neurones, qui reflètent la structure du génotype.

neurone. Chaque symbole DIV a deux descendants, qui peuvent prendre trois formes :

- deux symboles CELL, c'est-à-dire deux neurones ;
- deux symboles DIV, c'est-à-dire deux sous-modules ;
- un symbole DIV et une liste de clones, le clone effectuant une copie du sous-réseau correspondant au DIV « frère »

À chaque symbole DIV est associé une liste de connexions entre les neurones du sous-arbre correspondant. Un génotype de MENNAG est donc capable de décrire une hiérarchie de modules, chacun d'eux pouvant être répété. Pour chaque niveau de l'arbre, les connexions extra-modules, constituant les entrées et sorties du module, peuvent être identifiées en propageant la liste des connexions de la racine vers le symbole DIV.

### 6.4.3 *Hiérarchie et évolution modulaire incrémentale*

Du fait de la structure arborescente du génotype de MENNAG, il est aisé d'identifier les modules du génotype et d'en déduire des modules phénotypiques. La plus grande différence entre MENNAG et l'approche du chapitre 5 réside dans l'évolution de la modularité de la correspondance génotype-phénotype. En effet, les opérateurs de parcellation, d'intégration et de différentiation permettent de faire évoluer la modularité de cette correspondance en l'augmentant (via la parcellation et l'intégration) ou en la baissant (via la différentiation). L'évolution peut alors utiliser un module phénotypique, extrait *a posteriori*, pour créer un module génotypique. Cette procédure tente de garder une unité entre les modules phénotypiques, observés, et ceux du génotype.

Dans MENNAG, la correspondance ne peut prendre qu'un seul sens, du génotype vers le phénotype ; rien ne garantit que les modules phénotypiques ainsi obtenus soient équivalent à ceux obtenus par une analyse a posteriori. En caricaturant ce comportement et l'appliquant à l'homme, on peut considérer que rien n'empêche un morceau de la main d'être dans le même module que le nez. Cependant, l'importance de la correspondance des modules du phénotype avec ceux du génotype est tempérée par le peu d'influence qu'a la différentiation sur les performance des expériences du chapitre 5. Tout se passe comme si un module était d'abord isolé par parcellation, puis amélioré par la méthode multiobjectif. Rien ne garantit là non plus que le module une fois amélioré et inséré dans le réseau principal correspond toujours à un module phénotypique au sens de la division modulaire que nous avons utilisé. Cependant, la méthode multiobjectif garantie que ce module effectue la fonction pour laquelle il a été sélectionné. Il constitue donc indubitablement un module fonctionnel présent dans le phénotype, quelle que soit la topologie du réseau dans lequel il est inséré. En sélectionnant les modules de MENNAG, isolés à partir du génotype, à l'aide de la même méthode multiobjectif, on peut donc s'attendre à un comportement similaire : les modules effectueront les fonctions pour lesquels ils ont été sélectionnés et constitueront donc des modules fonc-

tionnels, même si rien ne garantit qu'une analyse du graphe les identifierait comme module.

La notion d'objectifs hiérarchiques qui seraient associés aux modules semble définir à définir. Cependant, il est possible qu'elle soit inutile car rien n'interdit à un objectif de tester une capacité de haut niveau nécessitant la présence de modules sélectionnés sur des objectifs de plus bas niveau. En un certain sens, tous les objectifs utilisés dans une démarche incrémentale sont mutuellement imbriqués.

## 6.5 BIAIS

Les méthodes évolutionnistes reposent sur la fonction de fitness pour diriger la recherche vers les bonnes solutions et sur les opérateurs génétiques pour explorer leur voisinage. La combinaison de ces deux éléments induit l'heuristique de recherche, idéalement indépendante du problème traité. Lorsque l'espace de recherche est très grand, par exemple lors de l'évolution de réseaux de neurones, il est possible d'orienter le processus vers certains types de solutions en modifiant la fonction de fitness, en modifiant les opérateurs ou en ajoutant des contraintes sur l'espace des solutions. Par conséquent, certaines solutions deviennent plus facile à obtenir que d'autres, biaisant ainsi la recherche.

En rendant accessibles des problèmes autrement inabordables, les biais constituent un élément très important pour le succès de beaucoup de méthodes d'évolution de réseaux de neurones. Malheureusement, ils sont souvent peu explicites, car induits par de nombreux éléments, difficiles à maîtriser, car enfouis dans les profondeurs des opérateurs et du génotype et généralement pas pris en compte lors des travaux comparant les performances des différentes méthodes d'évolution. Ce dernier point mérite d'être souligné car il signifie que la quantité de connaissances introduites dans le processus n'est que rarement considérée. Or, le *credo* de l'évolution de réseaux de neurone est justement la réduction des connaissances injectées par l'utilisateur dans le système. Ainsi, si l'on souhaite obtenir la preuve qu'une méthode d'évolution de réseau de neurones peut obtenir une solution efficace pour une tâche donnée, estimer la quantité d'information injectée dans le processus est critique pour distinguer la part de la solution « découverte » par l'évolution et celle apportée par l'expérimentateur.

L'omission des biais dans toute comparaison ne peut donc qu'être inéquitable. Plus de biais facilite l'obtention de solutions mais au prix d'une réduction de la généralité la méthode. L'utilisation de certains biais peut cependant être fondée, voire être une étape indispensable pour utiliser l'évolution pour des systèmes complexes. Il est certain que l'évolution naturelle exploite de nombreux biais, comme par exemple la géométrie et que ces biais participent à la réussite du processus évolutionniste. Il n'y a donc pas de raison de les éviter mais nécessité de les comprendre pour mieux les utiliser.

Dans cette section, nous commencerons par analyser les principaux biais utilisés par les méthodes de la littérature et par celles introduites dans cette thèse. Nous envisagerons d'abord ceux causés par la pression de sélection puis ceux induits par le codage. Nous proposerons ensuite quelques idées pour estimer les biais des méthodes d'évolution de réseaux de neurones.

### 6.5.1 *Pression de sélection*

L'approche la plus intuitive pour orienter la recherche est de modifier la fonction de fitness pour favoriser certaines solutions. Pour obtenir des réseaux avec le moins de connexions possibles, Ijspeert et al. (1999) ont ainsi multiplié la fitness par le rapport entre le nombre de connexions effectivement utilisées et leur nombre maximum possible. Le processus évolutionniste a ainsi été biaisé vers les solutions avec une faible connectivité, jugées plus intéressantes ou pouvant mener à de meilleures performances. Le *fitness shaping* (Nolfi et Parisi (1995), voir la section 2.4.3) s'appuie sur une technique proche. En modifiant la fitness de manière à ce que des comportements intermédiaires fassent augmenter la fitness, Nolfi et Parisi (1995) dirigent la recherche en poussant à l'émergence de solutions intermédiaires. L'utilisation d'une pondération entre les fitness correspondant à chaque comportement modifie aussi l'espace de recherche en définissant l'importance relative de chacun d'eux et donc l'ordre dans lequel ils doivent être trouvés. Comme nous l'avons vu dans la section 2.4.3, les méthodes d'évolution incrémentale par étape (Harvey et al., 1994; Gomez et Miikkulainen, 1997; Kodjabachian et Meyer, 1997; Urzelai et Floreano, 1999) induisent aussi de nombreux biais et contraintes arbitraires comme l'ordre des étapes, la nécessité de toutes les réussir et le moment du passage à l'étape suivante.

Les approches incrémentales et modulaires introduites dans cette thèse sont aussi une manière d'ajouter des biais pour diriger le processus de recherche. Même si, contrairement aux méthodes précédentes, l'ordre d'obtention des différentes sous-solutions n'est pas imposé, nous appliquons une pression de sélection incitant le processus évolutionniste à résoudre les sous-tâches, dirigeant ainsi l'évolution sur un chemin dont on fait l'hypothèse de son utilité. En n'imposant ni le chemin suivi ni la résolution de chaque sous-tâche, nous avons transformé une contrainte forte en un simple biais. En d'autres termes, nous avons remplacé une décision imposée par une suggestion.

### 6.5.2 *Biais du codage*

Au delà des modifications de la pression de sélection, trois approches peuvent être envisagées pour biaiser le processus d'évolution de réseaux de neurones :

– contraindre l'espace des solutions en se restreignant à certaines topologies, par exemple en interdisant les connexions récurrentes ;

– utiliser des opérateurs biaisés, par exemple en augmentant les chances de créer une connexion récurrente ou en permettant de dupliquer un sous-réseau ;
– commencer le processus évolutionniste à partir d'un nombre restreint et bien choisi de topologies ;

Le biais le plus utilisé pour l'évolution de réseaux de neurones est certainement d'orienter la recherche vers les topologies *feed-forward*. Dans la plupart des problèmes de pilotage de robots, des données issues de capteurs doivent être transformées pour mener à des commandes motrices. Dans les tâches de classification et d'approximation, une autre application typique des réseaux de neurones, les données d'entrée doivent être traitées pour obtenir une classe ou une valeur. Dans ces deux situations comme dans beaucoup d'autres, le réseau possède donc clairement un sens, allant des entrées vers les sorties. Le succès des applications des perceptrons multicouches montre l'utilité de tels réseaux *feed-forward* pour la classification et la modélisation de systèmes non linéaires (voir par exemple Haykin (1998)). Si des connexions récurrentes peuvent être utiles pour le pilotage de robots, par exemple pour calculer l'intégrale d'un signal ou implémenter une mémoire, elles restent une exception. NEAT (Stanley et Miikkulainen, 2002a), par exemple, implémente un tel biais en commençant l'évolution avec des réseaux *feed-forward* sans couches cachées. Il utilise ensuite des opérateurs de mutation ajoutant des neurones en divisant une connexion existante, ce qui préserve le sens du réseau, et en permettant d'interdire les connexions récurrentes. Cette orientation de la recherche est sans doute une des raisons du succès de NEAT pour les applications de contrôle comme le double pendule inversé (voir la section 2.5.1). De façon moins explicite, ModNet (Doncieux et Meyer, 2004b) utilise aussi un biais lors de la génération aléatoire.

Même augmentés de quelques connexions récurrentes, les réseaux *feed-forward* ne permettent pas de créer des CPG pour la locomotion car la dynamique rythmique de ces réseaux provient essentiellement de boucles. Les travaux sur les oscillateurs (Matsuoka, 1987; Chiel et al., 1999) mettent en jeu de nombreuses connections récurrentes. Biologiquement, les oscillateurs segmentaux de la lamproie (Ekeberg et al., 1995) s'appuient sur une topologie très éloignée de celle des réseaux *feed-forward*. Un autre type de biais doit donc être utilisé.

Au delà de leur structure hiérarchique, une caractéristique évidente de l'organisation des êtres vivants est l'influence de la géométrie des organismes sur l'organisation de leurs composants. Parce que la connexion de neurones distants est coûteuse à l'organisme, il existe une pression évolutionniste claire pour placer les neurones connectés entre eux aussi proches que possible. Ce principe, connu sous le nom de « principe d'optimisation des connexions »(*wiring optimization principle*) peut, par exemple, expliquer pourquoi le cortex visuel préserve l'organisation rétinotopique de l'oeil (Chklovskii et Koulakov, 2004). Les neurones proches de l'oreille droite, par exemple, reçoivent plus de

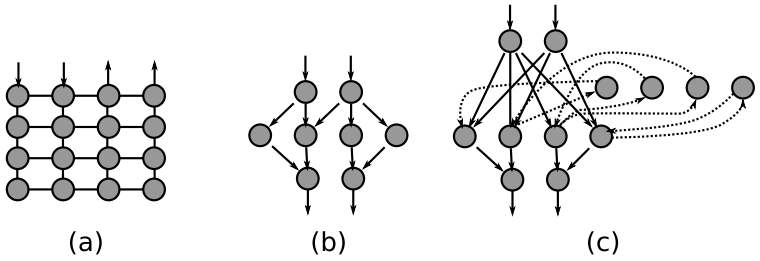



FIG. 6.8.: Exemples de réseaux de neurones cibles pour la robotique évolutionniste. (a) réseau complètement connecté. (b) Réseau *feed-forward*. (c) Réseau de Elman.

signaux de l'oreille droite que de l'oreille gauche ; l'ensemble des neurones traitant le signal auditif droit sont alors proches, formant naturellement un module et pouvant être plus facilement connectés entre eux. Symétriquement, les neurones reliés aux muscles sont placés en priorité près de ceux-ci, ce qui permet notamment des réflexes rapides. Les propriétés de l'espace physique biaisent ainsi l'organisation du réseau de neurones vers une connectivité locale utile pour résoudre les problèmes projetés du monde réel via les organes sensoriels. Ce biais autorise des topologies plus diverses que le biais vers les réseaux *feed-forward* mais permet quand même de tracer un chemin cohérent entre les entrées et sorties. Il a notamment été utilisé avec succès pour obtenir des neuro-contrôleurs pour un robot hexapode (Kodjabachian et Meyer, 1997). L'utilisation de ce type de biais pourrait avoir pour conséquence un biais vers des solutions modulaires (Pan et Sinha, 2007).

Les méthodes proposées dans cette thèse n'utilise ni de biais vers les réseaux *feed-forward*, ni de biais géométrique. Cependant, l'opérateur d'intégration utilisé pour l'approche modulaire change le voisinage d'une solution en rendant facilement accessible les réseaux utilisant plusieurs fois un même sous-réseau. Cette proximité est illustrée par la facilité que semble avoir le processus évolutionniste à trouver un réseau de neurones calculant la fonction booléenne $\big[(a \oplus b) \wedge (c \oplus d)\big]$ une fois qu'un module effectuant la fonction *xor* ($\oplus$) a été trouvé. L'opérateur d'intégration ajoute un biais vers les solutions modulaires et répétitives. En nous basant sur l'observation de la répétition des structures modulaires des êtres vivants et des artefacts complexes, nous avons traduit par l'emploi de ce biais l'hypothèse que des systèmes modulaires et répétitifs étaient de meilleurs candidats pour effectuer de nombreuses autres tâches.

### 6.5.3 *Expliciter et estimer les biais*

Malgré l'importance de l'utilisation des biais pour l'évolution de réseaux de neurones, nous ne disposons, à notre connaissance, d'aucune mesure de la quantité d'information injectée dans un système évolutionniste ni d'aucune

méthode de détermination et surtout de comparaison des biais. Ce manque empêche des comparaisons équitables entre méthodes et rend surtout difficile l'identification puis l'exploitation des biais potentiellement utiles à chaque domaine d'application. Si les biais causés par la pression de sélection sont généralement explicites, ceux induits par le codage sont souvent plus difficiles à cerner ; nous nous intéresserons donc essentiellement à ces derniers.

La mise au point du codage MENNAG (section 6.4.2) nous a fourni un exemple frappant de biais caché mais influant fortement sur les performances. Lors de tests, nous avons cherché à obtenir des réseaux pour des problèmes dont nous connaissions des solutions optimales. Nous avons alors constaté que le processus évolutionniste trouvait facilement des solutions très proches de celles recherchée mais n'arrivait presque jamais à les atteindre exactement. La solution se trouvait dans les liens entre l'opérateur de mutation et la structure modulaire et hiérarchique. Dans MENNAG, une liste de connexions est attachée à chaque module, chacune d'elles ne pouvant connecter que des neurones contenus dans ce module. L'opérateur de mutation peut modifier chaque connexion de trois façons : (1) changer la source, (2) changer sa cible et, (3) la supprimer. Il peut aussi ajouter à n'importe quel module une nouvelle connexion de source et de cible aléatoire. Lorsque l'évolution obtenait des réseaux proches de la solution optimale, une unique connexion restait à modifier pour pointer vers un autre neurone. Pour un codage direct, c'est une opération aisée mais pas dans la structure modulaire utilisée ici. Si la nouvelle cible de la connexion est située hors du module, ce qui était généralement le cas, la mutation doit d'abord enlever la connexion puis créer une nouvelle connexion au bon niveau entre les bons neurones. Cette opération a très peu de chances d'arriver en une unique génération. Or, la suppression d'une connexion dans cette situation faisait baisser la fitness ou ne l'augmentait pas. En conséquence, les solutions « sur la bonne voie » n'étaient pas sélectionnées et le processus restait coincé dans le minimum local. Cet exemple souligne l'importance de l'estimation expérimentale des biais, au moins pour la mise au point de nouveaux codages : la structure du génotype, dans notre cas la structure hiérarchique et modulaire, peut parfois rendre très éloigné une solution qui pourtant se trouve dans le voisinage du même réseau s'il est encodé avec un codage simple.

En poursuivant l'approche utilisée par MENNAG, il apparaît possible de mieux comprendre les biais liés au codage en tentant d'approcher des solutions aux caractéristiques données sous une pression de sélection la plus explicite possible. Un codage direct, incluant par définition peu de biais, peut servir de référence. Ainsi, on peut répondre à la question « Le codage X peut-il atteindre un réseau avec la caractéristique Y sous la pression de sélection "obtenir Y" ? ». De plus, l'analyse peut être affinée en comparant les temps de convergence de chaque méthode, donnant ainsi un indice sur la facilité d'obtention de la caractéristique étudiée. De nombreuses caractéristiques concernant la topologie des réseaux peuvent être étudiée, par exemple :

– son nombre de neurones ;

– sa modularité ;
– sa densité ;
– sa distance à un graphe cible.

Dans chacune de ces situations, une fitness visant à obtenir une valeur particulière peut être définie. Par exemple, on pourrait lancer une expérience pour obtenir des réseaux de densité 0.5 comportant 22 neurones. Nous devrions alors utiliser deux objectifs : (1) la différence entre la densité mesurée et la densité visée et (2) la différence entre le nombre de neurones du réseau et celle désirée. Une telle fitness n'est pas dénuée de biais mais elle demeure plus explicite que des fitness basées sur des expériences d'application robotique.

La distance à un graphe cible est probablement la caractéristique la plus intéressante. Obtenir un graphe que l'on sait capable de résoudre un problème permet de fournir des arguments pour répondre à la question « si la pression de sélection était idéale, ce codage permettrait-il de résoudre efficacement ce problème ? ». Dans la perspective de la robotique évolutionniste, nous avons identifié trois topologies utilisée dans de nombreux travaux qui pourraient constituer les premier graphes cibles (figure 6.8) :

– les réseaux complètement connectés ;
– les perceptrons multicouches avec une couche cachée ;
– les réseaux de Elman (Elman, 1990), permettant notamment de résoudre les problèmes nécessitant une mémoire.

Une fois ces réseaux choisis, l'étape suivante est d'utiliser comme fitness la distance d'édition entre les solutions candidates et le graphe cible. S'il est impossible d'utiliser une distance de graphe en raison de la complexité algorithmique d'une telle procédure, ces réseaux sont suffisamment particuliers pour autoriser la mise au point de distances *ad hoc* rapides à évaluer.

## 6.6 CONCLUSION

L'approche incrémentale et modulaire proposée dans cette thèse peut être utilisée pour de nombreux problèmes de robotique évolutionniste, de l'évolution de neuro-contrôleurs pour le pilotage de drones à celle de CPGs pour la locomotion rythmique. À titre d'exemple, des résultats préliminaires ont été obtenus pour un problème de panneaux de signalisation.

Du fait que chaque module ou chaque sous-tâche nécessite un objectif, nous sommes amenés à utiliser les algorithmes évolutionnistes avec beaucoup d'objectifs. Or, de nombreux auteurs ont constaté empiriquement que les performances de ces algorithmes se dégradaient fortement au-delà de trois objectifs. Ces conclusions ont été confirmées par une étude théorique montrant qu'ils ne peuvent pas être significativement plus performants qu'une recherche aléatoire si plus de trois objectifs *conflictuels* sont employés. Les objectifs que nous avons définis ne sont conflictuels que s'ils correspondent à plusieurs hypothèses en parallèle, nous pouvons donc envisager leur utilisation avec plus de trois objectifs. Afin d'augmenter les performances des algorithmes, il est

néanmoins possible d'essayer d'exploiter la forme particulière de l'espace objectif pour réduire le nombre d'objectifs. De plus, certains auteurs ont proposé des variantes de la domination de Pareto qui pourraient permettre d'optimiser plus d'objectifs.

La hiérarchie est une des caractéristiques importantes de l'architecture des systèmes complexes. Pour des raisons de simplicité, nous avons choisi de ne pas l'inclure dans ce travail afin de nous concentrer sur les relations entre modularité, génotype, phénotype et sélection. Nous avons d'abord noté que la notion de hiérarchie peut être peu pertinente compte tenu de la faible complexité par rapport au vivant des systèmes envisagés pour l'instant avec les méthodes évolutionnistes. Cependant, nous avons, dans des travaux récents, proposé un codage modulaire, hiérarchique et répétitif, formalisé avec une grammaire à attributs. Il pourrait être utilisé avec l'évolution modulaire incrémentale.

Les biais sont une caractéristique sous-jacente à toutes les méthodes d'évolution de réseau de neurones. Ils sont rarement analysés explicitement mais ont paradoxalement une grande influence sur les performances. Par exemple, des biais vers les réseaux *feed-forward* et d'autres exploitant la géométrie ont souvent permis une nette amélioration des performances. Dans le cadre de cette thèse, La méthode proposée est biaisée vers les réseaux modulaires et répétitifs. En outre, l'ajout des objectifs pour guider l'évolution peut aussi être vu comme un biais. Il apparaît important de pouvoir expliciter et évaluer ces biais afin de comprendre la quantité de connaissances injectées dans le processus évolutionniste. Une première idée dans ce sens peut être de mesurer les difficulté rencontrées par chaque méthode envisagée pour atteindre certains réseaux particuliers. Suivant les réseaux cible choisis, on peut définir des fitness rapides à évaluer et ainsi caractériser un codage indépendamment de tout problème applicatif.

# 7

# CONCLUSION

LA SÉLECTION NATURELLE forme la base des explications scientifiques de l'origine des espèces. Les lois de la variation héréditaire et l'ADN n'avaient pas été découvertes lors de la publication de l'*Origine des espèces* mais la simple hypothèse de leur existence combinée au concept de sélection naturelle suffit à Darwin pour formuler sa théorie et exhiber de nombreux exemples. En d'autres termes, comprendre l'essentiel des principes de l'apparition du vol des oiseaux ou la vitesse du guépard ne nécessite pas l'utilisation de caractéristiques précises du génome ; l'existence de « réplicateurs » et d'une force de sélection est suffisante (voir Dawkins (1976)).

La place centrale de la sélection naturelle en biologie évolutionniste contraste étonnamment avec celle qui lui est réservée dans les algorithmes évolutionnistes. Reléguée le plus souvent au statut de simple paramètre défini par l'utilisateur[1], elle ne peut rivaliser avec la richesse de la sélection naturelle. Les méthodes développées dans les précédents chapitres la replacent au centre des algorithmes évolutionnistes en montrant comment l'ajout de gradients de sélections peut modifier la difficulté des problèmes abordés.

Dans le premier chapitre, nous avons ainsi pu montrer que de tels gradients pouvaient fournir des étapes intermédiaires au processus évolutionniste. L'opportunisme de l'évolution peut l'amener à exploiter chaque gradient lorsque c'est possible et à s'appuyer sur les individus ainsi obtenus pour résoudre des tâches de plus en plus complexes. Les caractéristiques acquises sous la pression des objectifs les plus simples ont alors été *exaptées* pour aborder des problèmes plus complexes. La pertinence de cette approche a été illustrée sur un problème de robotique mobile. Difficile à aborder avec une fitness classique, il a été résolu en quelques centaines de générations lorsque de nouveaux gradients de sélection ont été ajoutés. L'addition des nouveaux objectifs a constitué un apport de connaissances au processus en le guidant vers des solutions. Néanmoins, en injectant les informations spécifiques à la tâche traitée au niveau de la sélection et non du codage, nous avons inversé le point de vue. La connaissance est alors utilisée plus explicitement tout en étant peu dépendante du type de « réplicateur » utilisé.

Le second chapitre expose une approche moins directive. En créant une pression de sélection favorisant la diversité des comportements, on crée implicitement un gradient de sélection pour chaque comportement possible, utile ou inutile à la résolution de la tâche visée. Un individu au comportement différent de celui de ses congénères sera sélectionné, incitant le reste de la population

1 Les expériences de co-évolution et d'évolution incrémentale constituent cependant des exemples notables de complexification du processus de sélection.

à le suivre. Nos expériences ont montré qu'une telle méthode pouvait mener à des résultats comparables à ceux obtenus en suivant des approches fondées sur des génotypes complexes. Une connaissance spécifique à la tâche est exploitée sous la forme d'une distance entre les comportements. Elle est explicite et dirige peu le processus évolutionniste.

Dans le troisième chapitre nous avons proposé d'ajouter des gradients de sélection en les associant à des modules de solutions. L'extraction des modules à partir du phénotype, sans se préoccuper du génotype, s'est révélée insuffisante ; tout comme l'utilisation d'un génotype modulaire sans lien explicite entre modules et sélection. Nos expériences ont montré que, dans la tâche envisagée, il était nécessaire de relier les modules du génotype, du phénotype et les gradients de sélection. En exploitant cet alignement, nous avons pu obtenir des solutions performantes en nettement moins de générations que les méthodes avec lesquelles nous avons comparé notre approche. Dans ces tests, les modules sélectionnés selon un gradient de sélection différent de l'objectif principal ont été employés par le processus évolutionniste pour mettre au point les solutions résolvant le problème principal. Il s'agit, par conséquent, d'un processus explicite d'exaptation, que nous avons pu diriger et contrôler. De plus, ces résultats expérimentaux mettent en évidence la nécessité d'employer un génotype modulaire pour pleinement bénéficier des pressions sélectives multiples. Ils réintroduisent donc dans notre réflexion les caractéristiques du génome, que nous avions laissées de côté dans les chapitres précédents, en insistant sur la nécessité de s'appuyer sur des correspondances génotype-phénotype modulaires.

Ces trois méthodes utilisent la relation de domination de Pareto pour incorporer des gradients de sélection aux algorithmes évolutionnistes. Cet outil, de plus en plus utilisé en optimisation, nous a permis de montrer l'intérêt de la multiplicité des gradients et certaines des conséquences que leur utilisation pouvait avoir sur les performances des méthodes évolutionnistes destinées à la robotique. Si les résultats décrits dans cette thèse montrent les bénéfices apportés par une telle approche multiobjectif à la robotique évolutionniste, ils soulignent surtout la nécessité de complexifier la notion de fitness pour pouvoir utiliser la puissance de l'évolution pour la synthèse de systèmes complexes.

# BIBLIOGRAPHIE


H. A. Abbass et K. Deb. Searching under multi-evolutionary pressures. *Proceedings of the Fourth Conference on Evolutionary Multi-Criterion Optimization, Spain*, 2003.

D. Ackley et M. Littman. Interactions between learning and evolution. *Artificial Life II*, 10 :487–507, 1992.

L. Altenberg. The evolution of evolvability in genetic programming. *Advances in Genetic Programming*, pages 47–74, 1994.

L. Altenberg. Modularity in Evolution : Some Low-Level Questions. In W. Callebaut et D. Rasskin-Gutman, editors, *Modularity : Understanding the Development and Evolution of Natural Complex Systems*. MIT Press, 2005.

L. W. Ancel et W. Fontana. Plasticity, evolvability, and modularity in RNA. *Journal of Experimental Zoology*, 288(3) :242–283, 2000.

J. R. Anderson. *How Can the Human Mind Occur in the Physical Universe ?*, chapter The Modular Organization of the Mind, pages 45–91. Oxford University Press, 2007.

P. J. Angeline, G. M. Saunders, et J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *Neural Networks, IEEE Transactions on*, 5(1) :54–65, Jan 1994.

M. Baatz et G. P. Wagner. Adaptive Inertia Caused by Hidden Pleiotropic Effects. *Theoretical Population Biology*, 51(1) :49–66, 1997.

T. Baeck, D. B. Fogel, et Z. Michalewicz. *Handbook of Evolutionary Computation*. Institute of Physics Publishing, 1997.

J. M. Baldwin. A new factor in evolution. *American Naturalist*, 30 :441–451, 1896.

R. Barate, S. Doncieux, et J.-A. Meyer. Design of a bio-inspired controller for dynamic soaring in a simulated UAV. *Bioinspiration & Biomimetics*, 1 :76–88, 2006.

W. Bateson. Materials for the Study of Variation. *Treated with Especial Regard to Discontinuity in the Origin of Species. London and New York : Macmillan Co*, 1894.

R. D. Beer, H. J. Chiel, et J. C. Gallagher. Evolution and analysis of model cpgs for walking : Ii. general principles and individual variability. *Journal of Computational Neuroscience*, 7(2) :119–147, 1999.

J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) :509–517, 1975.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

J. Blynel et D. Floreano. Exploring the T-maze : Evolving learning-like robot behaviors using CTRNNs. *Applications of Evolutionary Computing*, pages 593–604, 2003.

J.A. Bolker. Modularity in Development and Why It Matters to Evo-Devo 1. *Integrative and Comparative Biology*, 40(5) :770–776, 2000.

J. Bongard. Evolving modular genetic regulatory networks. *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, 2, 2002.

J. Bongard, V. Zykov, et H. Lipson. Resilient Machines Through Continuous Self-Modeling. *Science*, 314(5802) :1118–1121, 2006.

J. T. Bonner. *The Evolution of Complexity by Means of Natural Selection*. Princeton University Press, 1988.

E. Bornberg-Bauer et H. S. Chan. Modeling evolutionary landscapes : Mutational stability, topology, and superfunnels in sequence space. *Proc Natl Acad Sci US A*, 96(19) :10689–10694, 1999.

C. P. Bowers et J. A. Bullinaria. Embryological modelling of the evolution of neural architecture. In G. Bugmann A. Cangelosi et R. Borisyuk, editors, *Modeling Language, Cognition and Action*, pages 375–384, Singapore, 2005. World Scientific.

V. Braitenberg. *Vehicles*. MIT Press, Cambridge MA, 1984.

R. A. Brooks. Artificial life and real robots. *Proceedings of the First European Conference on Artificial Life*, pages 3–10, 1992.

A. Bucci, J. B. Pollack, et E. D. de Jong. Automated extraction of problem structure. In *GECCO (1)*, pages 501–512, 2004.

L.T. Bui, H. A. Abbass, et J. Branke. Multiobjective optimization for dynamic environments. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 3 : 2349–2356 Vol. 3, Sept. 2005.

H. Bunke. Recent developments in graph matching. *Proc. 15th International Conference on Pattern Recognition*, pages 117–124, 2000.

E. Burke, S. Gustafson, et G. Kendall. A survey and analysis of diversity measures in genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 716–723, 2002.

R. Calabretta, S. Nolfi, D. Parisi, et G. P. Wagner. A case study of the evolution of modularity : towards a bridge between evolutionary biology, artificial life, neuro- and cognitive science. In C. Adami, R. Belew, H. Kitano, et C. Taylor, editors, *Proceedings of Artificial Life VI*. MIT Press, 1998.

W. Callebaut et D. Rasskin-Gutman. *Modularity : Understanding The Development And Evolution Of Natural Complex Systems*. MIT Press, 2005.

A. Cangelosi, D. Parisi, et S. Nolfi. Cell division and migration in a 'genotype' for neural networks. *Network : Computation in Neural Systems*, 5(4) :497–515, 1994.

G. Capi et K. Doya. Evolution of Neural Architecture Fitting Environmental Dynamics. *Adaptive Behavior*, 13(1) :53, 2005.

S. B. Carroll. Chance and necessity : the evolution of morphological complexity and diversity. *Nature*, 409 :1102–1109, 2001.

W. E. Castle. Mendel's law of heridity. *Science*, 18(456) :396–406, 1903.

A. D. Channon et R. I. Damper. Evolving Novel Behaviors via Natural Selection. *Artificial Life VI : Proceedings of the Sixth International Conference on Artificial Life*, 1998.

J. Chavas, C. Corne, P. Horvai, J. Kodjabachian, et J.-A. Meyer. Incremental evolution of neural controllers for robust obstacle-avoidance in Khepera. In P. Husbands et J.-A. Meyer, editors, *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot98*, pages 227–247. Springer-Verlag, 1998.

J. M. Cheverud. Developmental Integration and the Evolution of Pleiotropy 1. *Integrative and Comparative Biology*, 36(1) :44–50, 1996.

H. J. Chiel, R. D. Beer, et J. C. Gallagher. Evolution and analysis of model cpgs for walking : I. dynamical modules. *Journal of Computational Neuroscience*, 7 (2) :99–118, 1999.

D. B. Chklovskii et A. A. Koulakov. Maps in the brain : What can We learn from them ? *Annual review of neuroscience*, 27 :369–392, 2004.

D. Cliff, I. Harvey, et P. Husbands. Incremental evolution of neural network architectures for adaptive behaviour. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN'93)*, pages 39–44, 1993a.

D. Cliff, P. Husbands, et I. Harvey. Explorations in Evolutionary Robotics. *Adaptive Behavior*, 2(1) :73, 1993b.

M. I. Coates et J. A. Clack. Polydactyly in the earliest known tetrapod limbs. *Nature*, 347(6288) :66–69, 1990.

C. A. C. Coello et G. B. Lamont. *Applications Of Multi-Objective Evolutionary Algorithms*. World Scientific, 2004.

E. D. Cope. The method of creation of organic forms. *Proceedings of the American Philosophical Society*, 12(86) :229–263, 1871.

D. W. Corne et J. D. Knowles. Techniques for highly multiobjective optimisation : some nondominated points are better than others. *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO)*, pages 773–780, 2007.

D. W. Corne, N. R. Jerram, J. D. Knowles, et M. J. Oates. PESA-II : Region-based Selection in Evolutionary Multiobjective Optimization. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 283–290, 2001.

F. Crick et J. Watson. A Structure for Deoxyribose Nucleic Acid. *Nature*, 171 (737-738), 1953.

D. B. D'Ambrosio et K. O. Stanley. A novel generative encoding for exploiting neural network sensor and output geometry. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007)*, 2007.

L. Danon, A. Diaz-Guilera, J. Duch, et A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics : Theory and Experiment*, 9 : P09008, 2005.

C. Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. 1st edition, 1859.

C. Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. 6th edition, 1872.

E. H. Davidson et D. H. Erwin. Gene Regulatory Networks and the Evolution of Animal Body Plans. *Science*, 311(5762) :796–800, 2006.

R. Dawkins. *The Selfish Gene*. Oxford, 1976.

R. Dawkins et J. R. Krebs. Arms Races between and within Species. *Proceedings of the Royal Society of London. Series B, Biological Sciences (1934-1990)*, 205(1161) :489–511, 1979.

H. P. O. de Beeck, J. Haushofer, et N. G. Kanwisher. Interpreting fMRI data : maps, modules and dimensions. *Nature Reviews Neuroscience*, 9 :123–135, 2008.

E. D. De Jong et A. Bucci. *Multi-Objective Problem Solving from Nature : From Concepts to Applications*, chapter Objective Set Compression : Test-based Problems and Multi-objective Optimization. Springer-Verlag, 2008.

E. D. de Jong et J. B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2), 2004a.

E. D. de Jong et J. B. Pollack. Ideal Evaluation from Coevolution. *Evolutionary Computation*, 12(2) :159–192, 2004b.

E. D. de Jong et D. Thierens. Exploiting modularity, hierarchy, and repetition in variable-length problems. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, 2004.

E. D. de Jong, R. A. Watson, et J. B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, 2001.

K. A. de Jong. *Evolutionary Computation*. The MIT Press, 2002.

E. de Margerie, J.-B. Mouret, S. Doncieux, J.-A. Meyer, T. Ravasi, P. Martinelli, et C. Grand. Flapping-wing flight in bird-sized UAVs for the ROBUR project : from an evolutionary optimization to a real flapping-wing mechanism. In *7th European Micro Air Vehicle Conference (MAV07)*, Toulouse, 2007.

R. De Nardi, J. Togelius, O.E. Holland, et S.M. Lucas. Evolution of neural networks for helicopter control : Why modularity matters. *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1799–1806, 0-0 2006.

K. Deb. *Multi-objectives optimization using evolutionnary algorithms*. Wiley, 2001.

K. Deb et D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the third international conference on Genetic algorithms*, pages 42–50, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. ISBN 1-55860-006-3.

K. Deb, S. Agrawal, A. Pratab, et T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization : NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, et Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000a. Springer. Lecture Notes in Computer Science No. 1917.

K. Deb, S. Agrawal, A. Pratap, et T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization : NSGA-II. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, 2000b.

K. Deb, M. Mohan, et S. Mishra. Evaluating the $\epsilon$-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evolutionary Computation*, 13(4) :501–525, 2005.

F. Delcomyn. Neural basis for rhythmic behaviour in animals. *Science*, 210 : 492–498, 1980.

A. Di Ferdinando, R. Calabretta, et D. Parisi. Evolving Modular Architectures for Neural Networks. *Connectionist Models of Learning, Development and Evolution : Proceedings of the Sixth Neural Computation and Psychology Workshop, Liege, Belgium, 16-18 September 2000*, 2001.

K. P. Dial. Wing-Assisted Incline Running and the Evolution of Flight. *Science*, 299(5605) :402–404, 2003.

S. Doncieux. *Evolution de Contrôleurs Neuronaux pour Animats Volants : Méthodologie et Applications*. PhD thesis, Université Paris 6, 2003.

S. Doncieux et J.-A. Meyer. Evolving neural networks for the control of a lenticular blimp. In G. R. Raidl, S. Cagnoni, J. J. Romero Cardalda, D. W. Corne, J. Gottlieb, A. Guillot, E. Hart, C. G. Johnson, E. Marchiori, J.-A. Meyer, et M. Middendorf, editors, *Applications of Evolutionary Computing, EvoWorkshops2003 : EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, EvoSTIM*. Springer Verlag, 2003a.

S. Doncieux et J.-A. Meyer. Evolving neural networks for the control of a lenticular blimp. In G. R. Raidl et al., editor, *Applications of Evolutionary Computing, EvoWorkshops2003 : EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, EvoSTIM*. Springer Verlag, 2003b.

S. Doncieux et J.-A. Meyer. Evolution of neurocontrollers for complex systems : alternatives to the incremental approach. In *Proceedings of The International Conference on Artificial Intelligence and Applications (AIA 2004)*, 2004a.

S. Doncieux et J.-A. Meyer. Evolving modular neural networks to solve challenging control problems. In *Proceedings of the Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004)*, 2004b.

S. Doncieux et J.-A. Meyer. Evolving PID-like neurocontrollers for non-linear control problems. *International Journal of Control and Intelligent Systems (IJCIS). Special Issue on nonlinear adaptive PID control*, 33(1) :55–62, 2005.

S. Doncieux, J.-B. Mouret, A. Angeli, R. Barate, J.-A. Meyer, et E. de Margerie. Building an artificial bird : Goals and accomplishments of the ROBUR project. In *Proceedings of the European Micro Aerial Vehicles (EMAV 2006) conference*, 2006.

S. Doncieux, J.-B. Mouret, et J.-A. Meyer. Soaring behaviors in UAVs : 'animat' design methodology and current results. In *7th European Micro Air Vehicle Conference (MAV07)*, Toulouse, 2007.

K. Doya et E. Uchibe. The Cyber Rodent Project : Exploration of Adaptive Mechanisms for Self-Preservation and Self-Reproduction. *Adaptive Behavior*, 13(2) :149, 2005.

P. Dürr, C. Mattiussi, et D. Floreano. Neuroevolution with analog genetic encoding. In Thomas Philip Runarsson, Hans-Georg Beyer, Edmund K. Burke, Juan J. Merelo Guervós, L. Darrell Whitley, et Xin Yao, editors, *PPSN*, volume 4193 of *Lecture Notes in Computer Science*, pages 671–680. Springer, 2006.

O. Ekeberg, A. Lansnser, et S. Grillner. The neural control of fish swimming studied through numerical simulations. *Adaptive Behavior*, 3 :363–384, 1995.

S. Elfwing, E. Uchibe, K. Doya, et HI Christensen. Biologically inspired embodied evolution of survival. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 3, 2005.

J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2) :179–211, 1990.

M. Farina et P. Amato. A fuzzy definition of "optimality" for many-criteria optimization problems. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 34(3) :315–326, 2004.

P. O. Fjallstrom. Algorithms for graph partitioning : A survey. *Linkoping Electronic Articles in Computer and Information Science*, 3(10), 1998.

D. Floreano et F. Mondada. Evolution of plastic neurocontrollers for situated agents. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, et S. Wilson, editors, *From Animals to Animats 4, Proceedings of the International Conference on Simulation of Adaptive Behavior*. MIT Press, 1996a.

D. Floreano et F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics–Part B : Cybernetics*, 26(3) :396–407, 1996b.

D. Floreano et J. Urzelai. Evolution of plastic control networks. *Autonomous Robots*, 11 :311–317, 2001.

D. Floreano, P. Dürr, et C. Mattiussi. Neuroevolution : from architectures to learning. *Evolutionary Intelligence*, 1(1) :47–62, 2008.

C. M. Fonseca et P. J. Fleming. Genetic algorithms for multiobjective optimization : formulation, discussion and generalization. In *Proceedings of the Fourth International Conference on Evolutionary Programming*, pages 416–423, 1993.

A. Force, M. Lynch, F. B. Pickett, A. Amores, Y. Yan, et J. Postlethwait. Preservation of Duplicate Genes by Complementary, Degenerative Mutations. *Genetics*, 151(4) :1531–1545, 1999.

A. Force, W. A. Cresko, F. B. Pickett, S. R. Proulx, C. Amemiya, et M. Lynch. The Origin of Subfunctions and Modular Gene Regulation. *Genetics*, 170(1) : 433–446, 2005.

I. Fukutani et J. Vaario. The effect of environment to genetic growth. In *International Symposium on System Life, July 21-22, 1997, Tokyo, Japan*, pages 227–232, 1997.

B. Fullmer et R. Miikkulainen. Using marker-based genetic encoding of neural networks to evolve finite-state behaviour. In *Proceedings of the first European Conference on Artificial Life (ECAL-91)*, Paris, 1991.

J. C. Gallagher, R. D. Beer, K. S. Espenschied, et R. D. Quinn. Application of evolved locomotion controllers to a hexapod robot. *Robotics and Autonomous Systems*, 19(1) :95–103, 1996.

N. García-Pedrajas, C. Hervás-Martínez, et J. Muñoz-Pérez. Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). *Neural Networks*, 15(10) :1259–1278, 2002.

N. Garcia-Pedrajas, C. Hervas-Martinez, et D. Ortiz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *Evolutionary Computation, IEEE Transactions on*, 9(3) :271–302, June 2005. ISSN 1089-778X. doi : 10.1109/TEVC.2005.844158.

J. J. Gauci et K. O. Stanley. Generating large-scale neural networks through discovering geometric regularities. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007)*, 2007.

W. Gerstner et W. Kistler. *Spiking Neuron Models : An Introduction*. Cambridge University Press New York, NY, USA, 2002.

S. Geva et J. Sitte. A cartpole experiment benchmark for trainable controllers. *IEEE Control Systems*, pages 40–51, 1993.

B. Girard, V. Cuzin, A. Guillot, K.N. Gurney, et T.J. Prescott. A basal ganglia inspired model of action selection evaluated in a robotic survival task. *Journal of Integrative Neuroscience*, 2(2) :179–200, 2003.

M. Girvan et M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12) :7821, 2002.

F. Glover, M. Laguna, et al. *Tabu search*. Springer, 1997.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

D. E. Goldberg et J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154, San Francisco, 1987. CA : Morgan Kaufmann.

F. Gomez et R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5 :317–342, 1997.

F. J. Gomez et R. Miikkulainen. Active guidance for a finless rocket using neuroevolution. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, 2003.

F. J. Gomez et R. Miikkulainen. Solving non-markovian control tasks with neuro-evolution. In *IJCAI '99 : Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1356–1361, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-613-0.

F. J. Gomez, J. Schmidhuber, et R. Miikkulainen. Efficient non-linear control through neuroevolution. In *Proceedings of the European Conference on Machine Learning (ECML-06, Berlin)*, pages 654–662, 2006.

S. J. Gould. Not Necessarily a Wing. *Natural History*, 94 :12–25, 1985.

S. J. Gould et E. S. Vrba. Exaptation ; a missing term in the science of form. *Paleobiology*, 8(1) :4–15, 1982.

W. K. Gregory. Reduplication in Evolution. *Quarterly Review of Biology*, 10(3) : 272, 1935.

M. Grönroos. A Comparison of Some Methods for Evolving Neural Networks. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2 :1442, 1999.

F. Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3(2) :151–183, 1995.

F. Gruau, D. Whitley, et L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In R. John Koza, E. David Goldberg, B. David Fogel, et L. Rick Riolo, editors, *Genetic Programming 1996 : Proceedings of the First Annual Conference*, pages 81–89, Stanford University, CA, USA, 28–31 1996. MIT Press.

G. Halder, P. Callaerts, et WJ Gehring. Induction of ectopic eyes by targeted expression of the eyeless gene in Drosophila. *Science*, 267(5205) :1788–1792, 1995.

N. Hansen et A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2) :159–195, 2001.

B. L. M. Happel et J. M. J. Murre. Design and evolution of modular neural network architectures. *Neural Networks*, 7(6-7) :985–1004, 1994a. ISSN 0893-6080. doi : http://dx.doi.org/10.1016/S0893-6080(05)80155-8.

L. M. Happel, B. et M. J. Murre, J. The design and evolution of modular neural network architectures. *Neural Networks*, 7 :985–1004, 1994b.

L. H. Hartwell, J. J. Hopfield, S. Leibler, et A. W. Murray. From molecular to modular cell biology. *Nature*, 402(6761) :C47–C52, 1999.

I. Harvey, P. Husbands, et D. Cliff. Seeing the light : artificial evolution ; real vision. In D. Cliff, P. Husbands, J.-A. Meyer, et S. Wilson, editors, *From Animals to Animats 3, Proceedings of the third international conference on Simulation of Adaptive Behavior*. MIT Press/Bradford Books, 1994.

I. Harvey, E. D. Paolo, R. Wood, M. Quinn, et E. Tuci. Evolutionary Robotics : A New Scientific Tool for Studying Cognition. *Artificial Life*, 11(1-2) :79–98, 2005.

S. Haykin. *Neural Networks : A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998. ISBN 0132733501.

W. D. Hillis. *Co-evolving parasites improve simulated evolution as an optimization procedure*, pages 228–234. MIT Press, Cambridge, MA, USA, 1991. ISBN 0-262-56057-7.

G. E. Hinton et S. J. Nowlan. How learning guides evolution. *Complex Systems*, 1 :495–502, 1987.

F. Hoffmann. Evolutionary algorithms for fuzzy control system design. *Proceedings of the IEEE*, 89(9) :1318–1333, 2001.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. MI : University of Michigan Press, 1975.

S. L. Hooper. Central pattern generators. In *Encyclopedia of Life Sciences*. Macmillan reference, 2001.

J. Horn, N. Nafpliotis, et DE Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 82–87, 1994.

G. S. Hornby. Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1729–1736, 2005.

G. S. Hornby et J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3) :223–246, 2002.

G. S. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, et M. Fujita. Evolving robust gaits with aibo. In *IEEE International Conference on Robotics and Automation*, pages 3040–3045, 2000.

G. S. Hornby, H. Lipson, et J. B. Pollack. Generative representations for the automated design of modular physical robots. *Robotics and Automation, IEEE Transactions on*, 19(4) :703–719, 2003.

S. A. Huettel, A. W. Song, et G. McCarthy. *Functional magnetic resonance imaging*. Sinauer Associates Sunderland, Mass, 2004.

T. S. Hussain. *Attribute Grammar Encoding of the Structure and Behaviour of Artificial Neural Networks*. PhD thesis, Queen's University, 2003.

C. Igel. Neuroevolution for reinforcement learning using evolution strategies. In *The 2003 Congress on Evolutionary Computation, CEC'03.*, volume 4, pages 2588–2595. IEEE Press, 2003.

A. J. Ijspeert et J. Kodjabachian. Evolution and development of a central pattern generator for the swimming of a lamprey. *Artificial Life*, 5(3) :247–269, 1999.

A. J. Ijspeert, J. Hallam, et D. Willshaw. Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Adaptive Behavior*, pages 151–172, 1999.

N. Jakobi. Evolutionary robotics and the radical envelop of noise hypothesis. *Adaptive Behavior*, 6(1) :131–174, 1997.

N. Jakobi. Running across the reality gap : Octopod locomotion evolved in a minimal simulation. In P. Husbands et J.-A. Meyer, editors, *Evolutionary Robotics. First European Workshop*, pages 39–58. Berlin : Springer Verlag, 1998.

F. Jiang, Berry H., et M. Schoenauer. Supervised and evolutionary learning of echo state networks. In *10th International Conference on Parallel Problem Solving From Nature, PPSN-2008*, 2008.

N. Kashtan et U. Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences*, 102(39) :13773–13778, 2005.

A. Keinan, B. Sandbank, C. C. Hilgetag, I. Meilijson, et E. Ruppin. Fair attribution of functional contribution in artificial and biological networks. *Neural Computuation*, 16(9) :1887–1915, 2004. ISSN 0899-7667.

V. R. Khare, X. Yao, et K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *EMO*, pages 376–390, 2003.

S. Kirkpatrick, C. D. .Jr. Gelatt, et M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4 :461–476, 1990.

J. D. Knowles et D. W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2) :149–172, 2000.

D. E. Knuth. Semantics of context-free languages. *Theory of Computing Systems*, 2(2) :127–145, 1968.

J. Kodjabachian et J.-A. Meyer. Evolution and development of control architectures in animats. *Robotics and Autonomous Systems*, 16 :161–182, 1995.

J. Kodjabachian et J.-A. Meyer. Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects. *IEEE Transactions on Neural Networks*, 9 :796–812, 1997.

J. Kodjabachian, C. Corne, et J. A. Meyer. Evolution of a robust obstacle avoidance behavior in khepera : A comparison of incremental and direct strategies. *Robotics and Autonomous Systems*, 1999. In Press.

J. B. Kollat et P. M. Reed. The Value of Online Adaptive Search : A Performance Comparison of NSGAII, $\varepsilon$-NSGAII and $\varepsilon$-MOEA. *The Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005). Lecture Notes in Computer Science*, 3410(386-398), 2005.

J. R. Koza. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

J. R. Koza et J. P. Rice. Genetic generation of both the weights and architecture for a neural network. *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, ii :397–404 vol.2, Jul 1991.

M. J. Walsh L. J. Fogel, A. J. Owens. *Artificial Intelligence Through Simulated Evolution*. Wiley, 1966.

C. G. Langton. *Artificial Life : An Overview*. Mit Press, 1995.

P. Larrañaga et J.A. Lozano. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

T. Larsen et ST Hansen. Evolving composite robot behaviour-a modular architecture. *Robot Motion and Control, 2005. RoMoCo'05. Proceedings of the Fifth International Workshop on*, pages 271–276, 2005.

M. Laumanns, L. Thiele, K. Deb, et E. Zitzler. Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. *Evolutionary Computation*, 10(3) :263–282, 2002.

J. Lehman et K. O. Stanley. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI)*, page to appear, 2008.

W. S. Levine. *The Control Handbook*. CRC Press, 1996.

M. A. Lewis, A. H. Fagg, et G. A. Bekey. Genetic algorithms for gait synthesis in a hexapod robot. In Y.F. Zheng, editor, *Recent trends in mobile robots*. World Scientific, 1993.

A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18(3) :280–315, 1968.

H. Lipson. Principles of Modularity, Regularity, and Hierarchy for Scalable Systems. *Genetic and Evolutionary Computation Conference (GECCO'04) Workshop on Modularity, regularity and Hierarchy*, 2004.

H. Lipson. Evolutionary Robotics and Open-Ended Design Automation. *Biomimetics : Biologically Inspired Technologies. Ed. Yoseph Bar-Cohen. Boca Raton, FL : CRC Press*, 2006.

H. Lipson et J.B. Pollack. Automaticdesign and manufactureofroboticlifeforms. *Nature*, 406(31) :974–978, 2000.

H. Lipson, J.B. Pollack, et N.P. Suh. On the origin of modular variation. *Evolution*, 56(8) :1549–1556, 2002.

P. López-Garcia et D. Moreira. Metabolic symbiosis at the origin of eukaryotes. *Trends in Biochemical Sciences*, 24(3) :88–93, 1999.

D. Lopresti et G. Wilfong. A fast technique for comparing graph representations with applications to performance evaluation. *International Journal on Document Analysis and Recognition*, 6(4) :219–229, 2003.

S. Luke et L. Spector. Evolving graphs and networks with edge encoding : Preliminary report. In *Late Breaking Papers of the Genetic Programming 96 (GP96)*, 1996.

E. Marder. Moving rhythms. *Nature*, 410 :755, 2001.

L. Margulis et R. Fester. *Symbiosis As a Source of Evolutionary Innovation : Speciation and Morphogenesis*. MIT Press, 1991.

K. Matsuoka. Mechanisms of frequency and pattern control in the neural rhythm generators. *Biological Cybernetics*, 56 :345–353, 1987.

C. Mattiussi et D. Floreano. Evolution of analog networks using local string alignment on highly reorganizable genomes. *Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on*, pages 30–37, 2004.

W. S. McCulloch et W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4) :115–133, 1943.

B. Meyer. *Object-Oriented Software Construction*. Prentice-Hall, 1997a.

J.-A. Meyer. Artificial life and the animat approach to artificial intelligence. In M. Boden, editor, *Artificial Intelligence*. Academic Press, 1996.

J.-A. Meyer. From natural to artificial life : Biomimetic mechanisms in animat designs. *Robotics and Autonomous Systems*, 22 :3–21, 1997b.

J.-A. Meyer. Evolutionary approaches to neural control in mobile robots. In *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, Piscataway, NJ, 1998. IEEE Press.

J.-A. Meyer, P. Husbands, et I. Harvey. Evolutionary robotics : a survey of applications and problems. In P. Husbands et J.-A. Meyer, editors, *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot98*. Springer Verlag, 1998.

J.-A. Meyer, S. Doncieux, D. Filliat, et A. Guillot. Evolutionary approaches to neural control of rolling, walking, swimming and flying animats or robots. In R. J. Duro, J. Santos, et M. Grana, editors, *Biologically Inspired Robot Behavior Engineering*. Springer Verlag, 2002.

J. G. Mezey, J. M. Cheverud, et G. P. Wagner. Is the Genotype-Phenotype Map Modular ? A Statistical Approach Using Mouse Quantitative Trait Loci Data. *Genetics*, 156(1) :305–311, 2000.

O. Miglino, H. Hautop L., et S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4) :417–434, 1995.

G. F. Miller, P. M. Todd, et S. U. Hedge. Designing neural networks using genetic algorithms. In *Proceedings of the Third International Conference on Artificial Intelligence*, pages 762–767. Morgan Kaufmann, 1989.

D. E. Moriarty. Symbiotic Evolution of Neural Networks in Sequential Decision Tasks. *University of Texas at Austin, Austin, TX*, 1998.

D. E. Moriarty et R. Miikkulainen. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5(4) :373–399, 1997.

A. Moshaiov. Multi-objective cybernetics and the concept-based approach : will they ever meet. *PPSN 2006 workshop on Multiobjective Problem Solving from Nature,(PPSN 2006)*, 2006.

J.-B. Mouret et S. Doncieux. Mennag : a modular, regular and hierarchical encoding for neural-networks based on attribute grammar. *Evolutionary Intelligence*, à paraître.

J.-B. Mouret et S. Doncieux. Incremental Evolution of Animats' Behaviors as a Multi-objective Optimization. In Y. Kuniyoshi, E. A. Di Paolo, G. Baldassarre, P. Husbands, J.-A. Meyer, J. Tani, S. Vijayakumar, et A. Philippides, editors, *From Animals to Animats 10*, 2008a.

J.-B. Mouret et S. Doncieux. Mennag : a modular, regular and hierarchical encoding for neural-networks based on attribute grammars. *Evolutionary Intelligence*, to appear, 2008b.

J.-B. Mouret, S. Doncieux, et J.-A. Meyer. Incremental evolution of target-following neuro-controllers for flapping-wing animats. In S. Nolfi, G. Baldassare, R. Calabretta, J.C. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, et D. Parisi, editors, *From Animals to Animats : Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB)*, pages 606–618, Rome, Italy, 2006.

G. B. Muller et G. P. Wagner. Novelty in Evolution : Restructuring the Concept. *Annual Reviews in Ecology and Systematics*, 22(1) :229–256, 1991.

M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3) :36104, 2006a.

M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23) :8577–8582, 2006b.

K. J. Niklas. *Plant Allometry : The Scaling of Form and Process*. University Of Chicago Press, 1994.

S. Nolfi et D. Floreano. *Evolutionary Robotics : Biology, Intelligence and Technology of Self-organizing Machines.* The MIT Press, 2001.

S. Nolfi et D. Floreano. Co-evolving predator and prey robots : Do 'arm races' arise in artificial evolution ? *Artificial Life*, 4(4) :311–335, 1998.

S. Nolfi et D. Parisi. Evolving non-Trivial Behaviors on Real Robots : an Autonomous Robot that Picks up Objects. *Topics in Artificial Intelligence : 4th Conference of the Italian Association for Artificial Intelligence, AI* IA'95, Florence, Italy, October 11-13, 1995 : Proceedings*, 1995.

S. Nolfi et D. Parisi. Learning to Adapt to Changing Environments in Evolving Neural Networks. *Adaptive Behavior*, 5(1) :75, 1996.

S. Nolfi et D. Parisi. "Genotypes" for neural networks. *The handbook of brain theory and neural networks*, pages 431–434, 1998.

S. Nolfi, J. L. Elman, et D. Parisi. Learning and evolution in neural networks. Technical report, Institute of psychology. C.N.R.- Rome, 1994.

N. NourAshrafoddin, A. R. Vahdat, et M. M. Ebadzadeh. Automatic Design of Modular Neural Networks Using Genetic Programming. *LECTURE NOTES IN COMPUTER SCIENCE*, 4668 :788, 2007.

J. Paakki. Attribute grammar paradigms : a high-level methodology in language implementation. *ACM Computing Surveys (CSUR)*, 27(2) :196–255, 1995.

K. Padian et L.M. Chiappe. The origin of birds and their flight. *Scientific American*, 278(2) :38–47, 1998.

R. K. Pan et S. Sinha. Modular networks emerge from multiconstraint optimization. *Physical Review E*, 76(4) :45103, 2007.

Z. Pan et L. Nie. Evolving both the topology and weights of neural networks. *International Journal of Parallel, Emergent and Distributed Systems*, 9(3) :299–307, 1996.

G.B. Parker. The Incremental Evolution of Gaits for Hexapod Robots. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 1114–1121, 2001.

F. Pasemann et U. Dieckmann. Evolved neurocontrollers for pole-balancing. In *Biological and Artificial Computation : From Neuroscience to Technology. IWANN'97*, 1997.

C. Paul et J. C. Bongard. The road less travelled : morphology in the optimization of bipedrobot locomotion. *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 1, 2001.

K. G. Pearson. Neural adaptation in the generation of rhytmic behavior. *Annual review of physiology*, 62 :723–753, 2000.

P. P. Pichler et L. Canamero. An Evolving Ecosystems Approach to Generating Complex Agent Behaviour. *Artificial Life, 2007. ALIFE'07. IEEE Symposium on*, pages 303–310, 2007.

M. A. Potter et K. A. De Jong. Cooperative Coevolution : An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8(1) :1–29, 2000.

K. Praditwong et X. Yao. How well do multi-objective evolutionary algorithms scale to large problems. *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3959–3966, 2007.

R. C. Purshouse et P. J. Fleming. On the Evolutionary Optimization of Many Conflicting Objectives. *Evolutionary Computation, IEEE Transactions on*, 11(6) : 770–784, 2007.

N. J. Radcliffe. Genetic set recombination and its application to neural network topology optimisation. *Neural computing and applications*, 1(1) :67–90, 1993.

T. S. Ray. An approach to the synthesis of life. *Artificial Life II*, 11 :371–408, 1991.

P. Reed, J. B. Kollat, et V. K. Devireddy. Using interactive archives in evolutionary multiobjective optimization : A case study for long-term groundwater monitoring design. *Environmental Modelling and Software*, 22(5) :683–692, 2007.

J. Reisinger et R. Miikkulainen. Acquiring evolvability through adaptive representations. *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1045–1052, 2007.

J. Reisinger, K. O. Stanley, et R. Miikkulainen. Evolving reusable neural modules. *Proc. of the Genetic and Evolutionary Computation Conference*, 2004.

R. Reveaux, B. Eugen, H. Locteau, S. Adam, P. Heroux, et E. Trupin. A Graph Classification Approach Using a Multi-objective Genetic Algorithm Application to Symbol Recognition. *Lecture notes in computer science*, 4538 :361, 2007.

S. H. Rice. The evolution of developmental interactions : epistasis, canalization, and integration. *Epistasis and the evolutionary process. Oxford Univ. Press, Oxford, UK*, pages 82–98, 2000.

C. D. Rosin et R. K. Belew. New Methods for Competitive Coevolution. *Evolutionary Computation*, 5(1) :1–29, 1997.

G. Rudolph. Convergence of evolutionary algorithms in general search spaces. *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 50–54, 1996.

G. Rudolph et A. Agapie. Convergence properties of some multi-objective evolutionaryalgorithms. *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, 2, 2000.

E. Ruppin. Evolutionary autonomous agents : a neuroscience perspective. *Nature Reviews Neuroscience*, 3(2) :132–141, 2002.

R. M. Rylatt et C. A. Czarnecki. Embedding Connectionist Autonomous Agents in Time : The 'Road Sign Problem'. *Neural Processing Letters*, 12(2) : 145–158, 2000.

G. Schlosser et G. P. Wagner. *Modularity in development and evolution*. Chicago : University of Chicago Press, 2004.

H. P. P. Schwefel. *Evolution and Optimum Seeking : The Sixth Generation*. John Wiley & Sons, Inc. New York, NY, USA, 1993.

J. D. Shaffer. *Some experiments in machine learning using vector evaluated genetic algorithms*. PhD thesis, Vanderblit University, 1985.

T. Shallice. *From Neuropsychology to Mental Structure*. Cambridge University Press, 1988.

J. Shan, C. Junshi, et C. Jiapin. Design of central pattern generator for humanoid robot walking based on multi-objective ga. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1930–1935, 2000.

L. S. Shapley. A value for n-person games. *Contributions to the theory of games*, 2 :307–317, 1953.

P. Shipman. *Taking Wing : Archaeopteryx and the Evolution of Bird Flight*. Simon & Schuster, 1998.

A. Siddiqi. Comparison of matrix rewriting versus direct encoding for evolving neural networks. *The 1998 IEEE International Conference on Evolutionary Computation, ICEC'98*, pages 392–397, 1998.

H. A. Simon. The Architecture of Complexity. *Proceedings of the American Philosophical Society*, 106(6) :467–482, 1962.

K. Sims. Evolving 3d morphology and behavior by competition. In R. A. Brooks et P. Maes, editors, *Proceedings of the Fourth International Workshop on Artificial Life*. The MIT Press, 1994.

S. Sorlin et C. Solnon. Reactive Tabu Search for Measuring Graph Similarity. *5th IAPR Workshop on Graph-based Representations in Pattern Recognition (GbR 2005)*, pages 172–182, 2005.

N. Srinivas et K. Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3) :221–248, 1994.

K. O. Stanley et R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2)(99-127), 2002a.

K. O. Stanley et R. Miikkulainen. Continual coevolution through complexification. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 113–120, 2002b.

K. O. Stanley et R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21(1) :63–100, 2004.

K. O. Stanley, B. D. Bryant, et R. Miikkulainen. Real-time neuroevolution in the NERO video game. *Evolutionary Computation, IEEE Transactions on*, 9(6) : 653–668, 2005.

S. C. Stearns. Trade-offs in life-history evolution. *Functional Ecology*, 3(3) :259–268, 1989.

R. S. Sutton et A. G. Barto. *Reinforcement Learning : An Introduction*. MIT Press, 1998.

C. Szyperski. *Component Oriented Programming*. Springer, 1998.

T. Taylor. Creativity in evolution : Individuals, interactions and environments. *Creative Evolutionary Systems*, pages 79–108, 2001.

S. A. Teichmann et M. M. Babu. Gene regulatory network growth by duplication. *Nature Genetics*, 36(5) :492–496, 2004.

J. Teo et H. A. Abbass. Coordination and synchronization of locomotion in a virtual robot. In L. Wang, J. C. Rajapakse, K. Fukushima, et X. Lee, S. Y. Yao, editors, *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP02)*, volume 4, pages 1931–1935, Singapore, 2002.

H. L. Teuber. Physiological Psychology. *Annual Review of Psychology*, 6(1) : 267–296, 1955.

O. Teytaud. How entropy theorems can show that offline approximating high-dim Pareto fronts is too hard. *PPSN BTP Workshop, PPSN*, 2006.

P. M. Todd et G. F. Miller. Exploring adaptive agency II : simulating the evolution of associative learning. *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats table of contents*, pages 306–315, 1991.

A. Toffolo et E. Benini. Genetic diversity as an objective in multi-objective evolutionary algorithms. *Evolutionary Computation*, 11(2) :151–167, 2003. ISSN 1063-6560.

P. D. Turney. A Simple Model of Unbounded Evolutionary Versatility as a Largest-Scale Trend in Organismal Evolution. *Artificial Life*, 6(2) :109–128, 2000.

J. Urzelai et D. Floreano. Incremental evolution with minimal resources. In *Proceedings of the First International Khepera Workshop*, 1999.

J. Urzelai, D. Floreano, M. Dorigo, et M. Colombetti. Incremental robot shaping. *Connection Science Journal*, 10(384) :341–360, 1998.

A. Wagner. Robustness, evolvability, and neutrality. *FEBS Letters*, 579(8) :1772–1778, 2005.

G. P. Wagner. Homologues, Natural Kinds and the Evolution of Modularity 1. *Integrative and Comparative Biology*, 36(1) :36–43, 1996.

G. P. Wagner et L. Altenberg. Complex adaptations and the evolution of evolvability. *Evolution*, 50(3) :967–976, 1996.

G. P. Wagner, J. Mezey, et R. Calabretta. Modularity : Understanding The Development And Evolution Of Natural Complex Systems. In W. Callebaut et D. Rasskin-Gutman, editors, *Modularity : Understanding the Development and Evolution of Natural Complex Systems*, chapter Natural Selection and the Origin of Modules. MIT Press, 2005.

T. Wagner, N. Beume, et B. Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, et T. Murata, editors, *EMO*, volume 4403 of *Lecture Notes in Computer Science*, pages 742–756. Springer, 2006.

J. Walker, S. Garrett, et M. Wilson. Evolving Controllers for Real Robots : A Survey of the Literature. *Adaptive Behavior*, 11(3) :179, 2003.

M. Walker. Comparing the performance of incremental evolution to direct evolution. *2nd International Conference on Autonomous Robots and Agents*, 2004.

S. Wasserman et K. Faust. *Social Network Analysis : Methods and Applications*. Cambridge University Press, 1994.

R. A. Watson. Modular interdependency in complex dynamical systems. *Artificial Life*, 11(4) :445–457, 2005.

D. Waxman et J. R. Peck. Pleiotropy and the Preservation of Perfection. *Science*, 279(5354) :1210, 1998.

H. C. White, S. A. Boorman, et R. L. Breiger. Social Structure from Multiple Networks. I. Blockmodels of Roles and Positions. *American Journal of Sociology*, 81(4) :730, 1976.

A. Wieland. Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks* (Seattle, WA), pages 667–673. Piscataway, NJ : IEEE, 1991.

L. Yaeger. Computational Genetics, Physiology, Metabolism, Neural Systems, Learning, Vision, and Behavior or Poly World : Life in a New Context. In *Sante Fe Institute studies in the sciences of complexity*, volume 17, pages 263–263. Addison-Wesley, 1994.

L. S. Yaeger et O. Sporns. Evolution of neural structure and complexity in a computational ecology. *Artificial Life, 2008. ALIFE'08. IEEE Symposium on*, 2008.

B. M. Yamauchi et R. D. Beer. Sequential Behavior and Learning in Evolved Dynamical Neural Networks. *Adaptive Behavior*, 2(3) :219, 1994.

X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9) :1423–1447, 1999.

C. H. Yuh. Modular cis-regulatory organization of Endo16, a gut-specific gene of the sea urchin embryo, 1996.

H. Zhu, P. A. V. Hall, et J. H. R. May. Software Unit Test Coverage and Adequacy. *ACM Computing Surveys*, 29(4), 1997.

T. Ziemke et M. Thieme. Neuromodulation of reactive sensorimotor mappings as a short-term memory mechanism in delayed response tasks. *Adaptive Behavior*, 10(3/4) :185–199, 2002.

E. Zitzler et L. Thiele. Multiobjective evolutionary algorithms : a comparative case studyand the strength Pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3(4) :257–271, 1999.

E. Zitzler, K. Deb, et L. Thiele. Comparison of Multiobjective Evolutionary Algorithms : Empirical Results. *Evolutionary Computation*, 8(2) :173–195, 2000.

E. Zitzler, M. Laumanns, L. Thiele, et al. SPEA2 : Improving the Strength Pareto Evolutionary Algorithm. *EUROGEN*, pages 95–100, 2001.

V. Zykov, J. Bongard, et H. Lipson. Evolving Dynamic Gaits on a Physical Robot. *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO*, 4, 2004.

V. Zykov, E. Mytilinaios, B. Adams, et H. Lipson. Self-reproducing machines. *Nature*, 435(7039) :163–164, 2005.

# A

# TESTS STATISTIQUES ET PARAMÈTRES

## A.1 T-TESTS

### A.1.1 *Évolution incrémentale modulaire*

| | Diversité | Modularité | Mod. + div. | NEAT |
|---|---|---|---|---|
| Gén. moyenne | 921.7 | 409.0 | 171.1 | 757.3 |
| Écart type | 463.1 | 224.7 | 66.16 | 499.4 |
| T-test Div. | $p = 1.0$ | $p < 0.001$ | $p < 0.001$ | $p = 0.3025$ |
| T-test Mod. | $p < 0.001$ | $p = 1.0$ | $p < 0.001$ | $p = 0.0022$ |
| T-test Mod+Div | $p < 0.001$ | $p < 0.001$ | $p = 1.0$ | $p < 0.001$ |
| T-test NEAT | $p = 0.3025$ | $p = 0.0022$ | $p < 0.001$ | $p = 1.0$ |

## A.2 PARAMÈTRES

### A.2.1 *Évolution incrémentale*

- MOEA : NSGA-II
- Taille de la population : 200
- Fitness : robot phototrope
- Réseaux de neurones (perceptron multicouche, structure fixe) :
  - nombre d'entrées : 15
  - nombre de sorties : 2
  - nombre de couches cachées : 1
  - poids minimum : $-10.0$
  - poids maximum : 10.0
  - type de mutation : gaussienne
  - proba. de mutation : 0.3
  - écart type (gaussienne) : 0.25
  - fonction d'activation : $\begin{cases} \tanh(\sum_j w_{ij} x_j) \text{ si } \sum_j w_{ij} x_j > 0 \\ 0 \text{ sinon} \end{cases}$

### A.2.2 *Diversité comportementale*

*Expérience* $\left[(a \oplus b) \wedge (c \oplus d)\right]$

- MOEA : NSGA-II
- Taille de la population : 400
- Fitness : $\left[(a \oplus b) \wedge (c \oplus d)\right]$

- Réseaux de neurones (codage direct) :
  - poids / biais possibles : $\{-2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0\}$
  - nombre d'entrées : 4
  - nombre de sorties : 1
  - min. neurones (gén. aléatoire) : 10
  - max. neurones (gén. aléatoire) : 20
  - min. connexions (gén. aléatoire) : 20
  - max. connexions (gén. aléatoire) : 35
  - proba. d'ajout de connexion : 0.15
  - proba. de suppression de connexion : 0.25
  - proba. de changement de connexion : 0.1
  - proba. d'ajout de neurones : 0.025
  - proba. de supression de neurone : 0.025
  - fonction d'activation : $y_i = \varphi\left(\sum_j w_{ij} x_j\right)$ avec $\varphi(x) = \frac{1}{1+\exp(b-kx)}$

*Robot phototrope*

- MOEA : NSGA-II
- Taille de la population : 200
- Fitness : robot phototrope
- Réseaux de neurones (perceptron multicouche, structure fixe) :
  - nombre d'entrées : 15
  - nombre de sorties : 2
  - nombre de couches cachées : 1
  - poids minimum : $-10.0$
  - poids maximum : 10.0
  - type de mutation : gaussienne
  - proba. de mutation : 0.3
  - écart type (gaussienne) : 0.25
  - fonction d'activation : $\begin{cases} \tanh(\sum_j w_{ij} x_j) \text{ si } \sum_j w_{ij} x_j > 0 \\ 0 \text{ sinon} \end{cases}$

### A.2.3 *Évolution incrémentale modulaire*

*Expérience liminaire*

- MOEA : NSGA-II
- Taille de la population : 400
- Fitness : $\left[(a \oplus b) \wedge (c \oplus d)\right]$
- Réseaux de neurones (codage direct) :
  - poids / biais possibles : $\{-2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0\}$
  - nombre d'entrées : 4
  - nombre de sorties : 1
  - min. neurones (gén. aléatoire) : 10
  - max. neurones (gén. aléatoire) : 20

  - min. connexions (gén. aléatoire) : 20
  - max. connexions (gén. aléatoire) : 35
  - proba. d'ajout de connexion : 0.15
  - proba. de suppression de connexion : 0.25
  - proba. de changement de connexion : 0.1
  - proba. d'ajout de neurones : 0.025
  - proba. de supression de neurone : 0.025
  - fonction d'activation : $y_i = \varphi\left(\sum_j w_{ij} x_j\right)$ avec $\varphi(x) = \frac{1}{1+\exp(b-kx)}$

*Expérience* $\left[(a \oplus b) \wedge (c \oplus d)\right]$

- MOEA : NSGA-II
- Taille de la population : 400
- Fitness : $\left[(a \oplus b) \wedge (c \oplus d)\right]$
- Réseaux de neurones (codage modulaire) :
  - poids / biais possibles : $\left\{-2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0\right\}$
  - nombre d'entrées : 4
  - nombre de sorties : 1
  - min. neurones (gén. aléatoire) : 10
  - max. neurones (gén. aléatoire) : 20
  - min. connexions (gén. aléatoire) : 20
  - max. connexions (gén. aléatoire) : 35
  - proba. de croisement (si utilisé) : 0.5
  - proba. de parcellation (si utilisé) : 0.25
  - proba. de différentiation (si utilisé) : 0.02
  - proba. d'intégration (si utilisé) : 0.1
  - proba. d'ajout de connexion : 0.15
  - proba. de suppression de connexion : 0.25
  - proba. de changement de connexion : 0.1
  - proba. d'ajout de neurones : 0.025
  - proba. de supression de neurone : 0.025
  - fonction d'activation : $y_i = \varphi\left(\sum_j w_{ij} x_j\right)$ avec $\varphi(x) = \frac{1}{1+\exp(b-kx)}$

*Panneaux de signalisation*

- MOEA : NSGA-II
- Taille de la population : 200
- Fitness : panneaux de signalisation
- Réseaux de neurones (codage modulaire) :
  - poids / biais possibles : $\left\{-2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0\right\}$
  - nombre d'entrées : 12
  - nombre de sorties : 2
  - min. neurones (gén. aléatoire) : 10
  - max. neurones (gén. aléatoire) : 20

- min. connexions (gén. aléatoire) : 20
- max. connexions (gén. aléatoire) : 35
- proba. de croisement : 0.5
- proba. de parcellation : 0.25
- proba. de différentiation : 0.05
- proba. d'intégration : 0.1
- proba. d'ajout de connexion : 0.15
- proba. de suppression de connexion : 0.25
- proba. de changement de connexion : 0.1
- proba. d'ajout de neurones : 0.025
- proba. de supression de neurone : 0.025
- fonction d'activation : $\begin{cases} \tanh(\sum_j w_{ij} x_j) \text{ si } \sum_j w_{ij} x_j > 0 \\ 0 \text{ sinon} \end{cases}$